제2권 통권 8호 1989년 7월 1일 발행 (매월 1일 발행) 1988년 10월 10일 등록 (등록번호 라-3794) ㉾ 140-150 서울특별시 용산구 갈월동 17

하이테크 퍼스널컴퓨터

89/07

특집 / CP/M-86의 분석

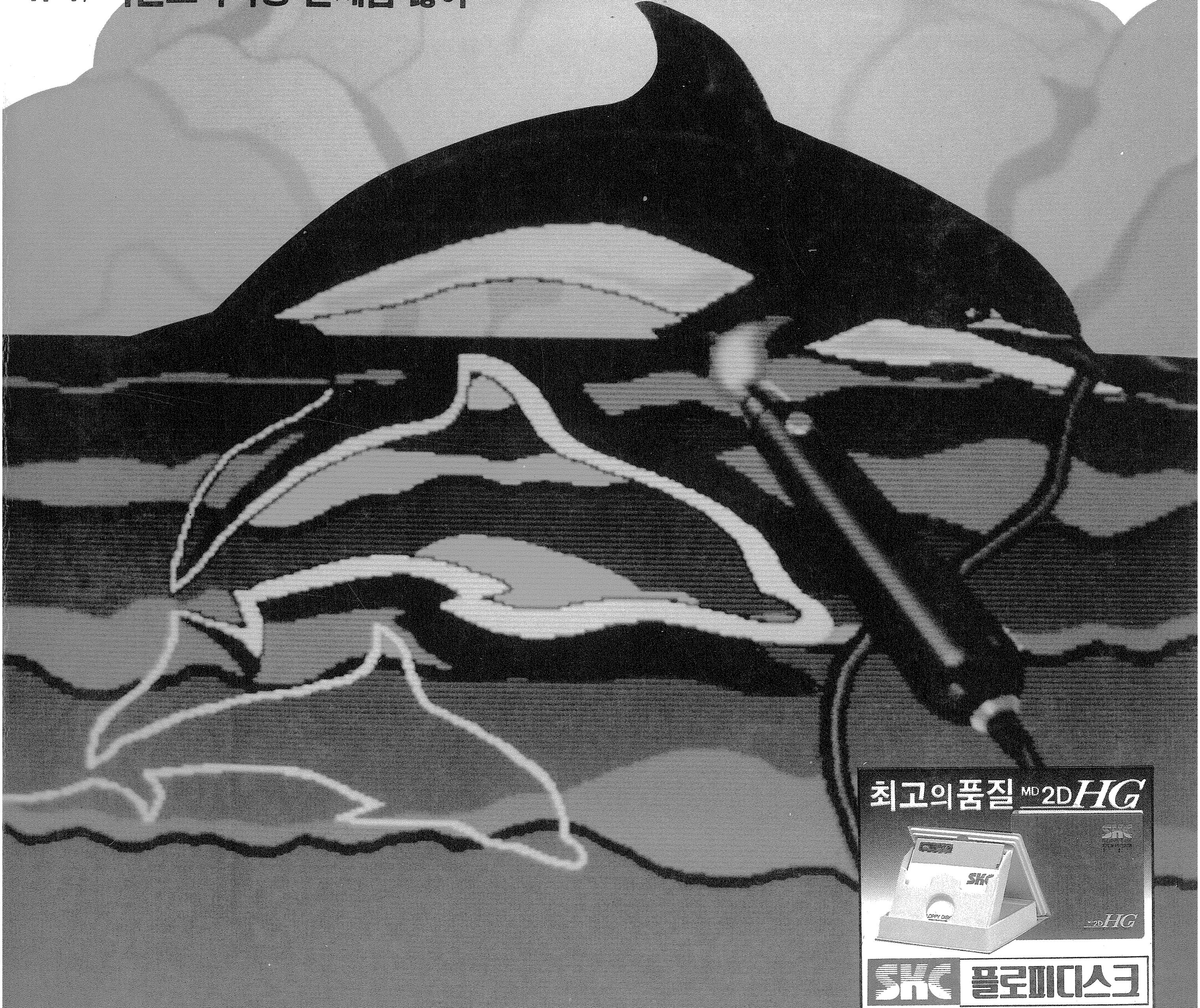
유틸리티 소개 / 강력한 도스쉘 기능의 1DIR

테크닉 / SEM-DRAW에 이미지 저장기능을 / NKP용 도스 유틸리티 / 인터럽트 핸들러

활용 / 프로시듀어 화일의 이용 / 기계어코드를 간단하게 베이직 데이터로 / 행렬을 이용한 이미지 변환

제작코너 / MCS-51 트레이닝 보드

취재 / 학원교육과정 문제점 많아



□ 편집자 노트 □

우리를 기쁘게 하는 것들

최근들어 저희를 즐겁게 하는 것이 두가지가 있습니다.
하나는 컴퓨터 주변기기 값들이 하루가 다르게 내려 누구나가 부담을 별로 가지지 않고 구입할 수 있게 된 것과 다른 하나는 저희 잡지의 정기구독자가 부쩍 늘었다는 사실입니다.
얼마전까지만 해도 PC-XT 1대를 구입하는데 드는 비용이 100만원대를 호가하였습니다.
그러던 것이 최근에는 대기업들이 앞을 다투어 40~50만원대의 보급기종을 내놓더니 이제는 50만원대 전후면 쓸만한 것을 구입할 수 있게 되었습니다.
이렇게 가격이 떨어지게 된 것에는 여러가지 이유가 있습니다.
원가절감을 위해 생산공정을 단축시켰다든지 아니면 부품가격이 많이 내렸다든지 혹은 카스텀 IC등을 사용하여 부품 수를 대폭 줄였다든지 등등.
그러나 이러한 여러가지 이유보다도 가장 큰 원인은 유저가 기하급수적으로 늘어 대중화 단계에 들어서서 대량생산이 가능했기 때문이 아닌가 생각됩니다.
컴퓨터 주변장치 중에서도 가장 많이 사용되는 디스크 드라이브는 컴퓨터 사용자에게 반드시 필요한 장치이나 가격이 비싸 추가구입시에는 사용자에게 많은 부담을 안겨 주었습니다. 이중 플로피 디스크 드라이브는 2년전만 해도 I/F카드 포함해서 20만원이상씩 하였습니다. 그러던 것이 현재는 I/F카드 포함해서 절반가격이면 구입이 가능하게 되었습니다. 또, 하드 디스크의 경우 1MB당 가격이 2년전에는 2만원정도 하던 것이 현재는 1만원 이하로 이것 역시 절반정도로 떨어졌습니다.
이들은 앞으로도 계속해서 가격이 떨어지리라고 생각됩니다.
심지어 교육용 PC의 경우 납품가격이 40만원 정도인 것으로 알려지고 있습니다.
따라서 앞으로도 계속해서 가격이 떨어진다면 지금의 가전제품처럼 각 가정에서 1대씩 구입하는 것도 가능하리라고 생각됩니다.
아무튼 이러한 사실들이 저희들을 매우 흐뭇하게 합니다. 누구나가 컴퓨터를 다룰 수 있고 또 그러한 시대를 추구하는 것이 저희 잡지가 가장 바라는 일이기 때문입니다.
컴퓨터 가격이 내리고 또 각 개인들이 컴퓨터를 많이 구입하여 보유함에 따라 저희 잡지의 구독자도 하루가 다르게 부쩍부쩍 늘어나는 추세입니다.
이러한 두가지 사실 때문에 저희들은 매우 흐뭇하며 신바람이 납니다.
무슨 일이라도 독자들에게 도움이 되는 일을 할 수 있다는 사실이 저희들의 가슴을 마냥 부풀게 합니다.
독자 여러분, 이 기회에 컴퓨터를 구입하여 열심히 배우십시요.
저희들이 최선을 다해 도와드리겠습니다.

유용한 유틸리티 3
록스미스

김원준 / 건국대학교 전자공학과

록스미스(Lock Smith)는 디스크 매니저(관리 프로그램)이자, 강력한 카피 유틸리티이다. 이 프로그램은 프로텍트된 디스크의 복사를 가능하게 해주며, 디스크의 내용을 직접 변경할 수 있도록 해준다. 또한 PCTOOLS 등이 가지고 있는 디스크 매니저 기능을 포함하고, 개인적인 화일의 보호를 위해 화일을 암호화할 수 있으며, 다른 카피 프로그램과 달리 한번 읽은 디스크의 내용을 여러장의 디스켓에 반복하여 복사할 수도 있다.

이와 같이 다양한 기능을 가진 록스미스이지만, 대부분의 사용자들은 단순히 카피용으로만 사용하고 있을 것이다. 그 이유는 PCTOOLS이나 노턴 유틸리티와 같은 사용하기 편리하고 강력한 프로그램들이 많이 있기 때문이기도 하지만 록스미스의 섬세한 기능들을 사용하기 위해서는 디스크에 대한 어느정도 심도있는 지식이 필요할 만큼 사용법이 어렵기 때문일 것이다.

본고에서는 록스미스의 기능들을 아주 전문적인 부분을 제외하고는 모두 소개해 보려 한다. 디스크에 대한 지식이 부족하다고 생각한다면, 다른 참고 서적을 통해 트랙이나 섹터 등에 대해 먼저 공부할 것을 권한다.

이 프로그램의 기능을 설명하기 전에 소프트웨어의 복제에 관한 문제를 짚고 넘어 가야 할 것 같다. 아직까지 우리나라는 불법 복사의 천국이라고 할 수 있을 정도로 무단 복제가 성행하고 있다. 아무리 강력한 프로텍트가 걸린 프로그램이 나와도 곧 무단으로 복제되어 시중에 나돌고 있는 것이다.

우리나라에 처음 컴퓨터가 소개 된 이후, 이 무단 복제가 우리나라 컴퓨터 문화의 발전에 많은 도움을 준 것은 사실이다. 외국에서 들여온 고가의 프로그램들을 우리는 무단 복제를 통해 손쉽게 얻어서 사용해 왔다. 하지만 이제는 지금까지 우리가 누려왔던 무단 복제의 혜택(?)에 대하여 다시 생각해야만 할 때가 되었다.

우리가 사용하고 있는 대부분의 프로그램들은 외국에서 개발된 것들이다. 이러한 상황을 보면 우리의 손으로 만들어진 소프트웨어의 필요성이 절실해진다. 더우기 우리가 많은 프로그램들을 사용하면서 가장 어려움을 느끼는 한글문제를 해결하는 것은 바로 우리의 손에 의해서 이루어져야 할 것이다.

하지만 많은 투자를 통해 개발된 소프트웨어가 불법 복사에 의해 누출된다면 아무도 소프트웨어 산업에 투자하려 하지 않을 것이다.

그러나 이러한 불법 복사문제는 비단 우리나라에만 국한된 것이 아니다. 소프트웨어의 저작권이 보장된다는 여러나라들에서도 복사는 이루어지고 있고, 록스미스와 같은 프로텍트된 프로그램을 복사해 주는 프로그램들이 버젓이 판매되고 있는 것을 보면 알 수 있다.

많은 소프트웨어 회사에서는 구입한 제품의 복사판을 만들어서 사용하고 원본은 손상되지 않도록 보관할 것을 권한다. 그러므로 자신이 구입한 프로그램의 손상을 방지하기 위해 복사판을 만들때에 사용되는 것이 이와 같은 카피 프로그램들이다. 이러한 맥락에서 본다면 카피프로그램의 사용은 이해될 수 있다.

사용가능한 시스템

록스미스를 사용하기 위한 시스템은 다음과 같다.

- IBM PC / XT 및 호환기종
- DOS를 제외한 128K 이상의 메모리
- 하나 이상의 40트랙 디스크 드라이브

그러므로 국내에 보급된 XT호환기종이면 대부분 사용이 가능하다.

LSF.EXE는 하드디스크를 대상으로 사용할 수 있으며, AT 호환기종에서도 사용이 가능하다. 록스미스는 LS.EXE, LSF.EXE, LSK1.EXE라는 세개의 화일로 구성되어 있다. LS.EXE는 록스미스의 디스크 버전이고, LSF.EXE는 화일 버전이다. LSK1.EXE는 LS.EXE로 복사한 특수한 프로텍트가 걸린 프로그램을 실행시킬 때 띄워놓고 쓰는 프로그램이다.

록스미스는 화면을 몇개의 윈도우로 나누어 각 윈도우마다 고유의 기능을 가지도록하여 체계적으로 동작하고 있다. 그러면 이제 부터는 각 화일별로 알아보자.

LS.EXE (디스크 버전)

이 프로그램은 디스크를 조작하는

프로그램이다. 이 프로그램을 사용하면, 특정 트랙, 섹터의 내용을 자유롭게 변경할 수 있고, 프로텍트된 디스크의 복사도 가능하다. 또한 다른 카피 프로그램에서는 찾아 보기 힘든 반복 복사의 기능을 가지고 있어서, 한번의 리드로 여러장의 복사판을 만들 수 있다(옵션기능 참조).

이 프로그램의 사용 방법은 크게 두가지로 나눌 수 있다. 한가지는 DOS 프롬프트상에서 LS 뒤에 옵션을 붙여 사용하는 것으로, 이 방법은 디스크의 복사만을 위해서 사용된다. 또 한 가지는 LS 프로그램 내에서 화면상단의 메뉴를 Cursor Driven 형식으로 사용하는 것이다. 이 방식으로는 디스크의 복사를 포함한 록스미스의 모든 기능을 사용할 수 있다.

(1) LS.EXE에서의 일반적인 규칙

•작업화면의 사용

록스미스의 디스크 버전(LS.EXE)은 화면을 그림 1에서와 같이 4개의 윈도우로 분할하여 사용한다. 각 윈도우의 기능들은 다음과 같다. 이 부분의 내용은 아직 다른 기능에 대하여 모르는 상태에서 완전히 이해하기는 쉽지 않을 것이므로 우선 가볍게 읽어 본 후 다른 내용들을 읽어 가면서 때때로 참고로 하면 될 것이다.

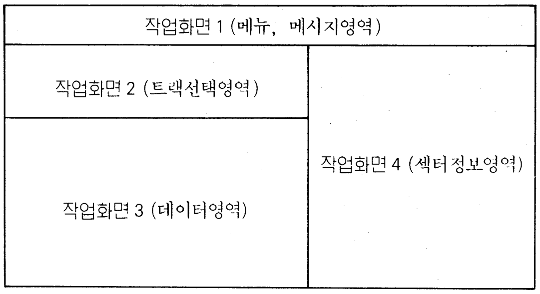
① 작업화면 1

이 영역은 각 메뉴의 선택을 위해 사용된다. 원하는 기능의 수행을 위해 화살표 키로 이동하여 [RETURN] 혹은 [ENTER] 키를 누르거나 메뉴의 첫 문자를 입력하면 선택된다. 또한 이 영역은 각종 기능의 수행 확인이나 각종 경고, 에러 메시지 등이 표시되는 부분이다.

② 작업화면 2

이 영역은 트랙 고르기 영역이다. LS.EXE는 원하는 트랙의 원하는 섹터만을 대상으로 작업하는 것이 가능하다. 그러므로 작업하고자 하는 트랙을 선택하고자 할때 이 작업화면을 사용한다.

〔그림 1〕 작업화면의 분할



다. 메뉴상태에서 [F1]키를 누르면 이 작업화면으로 커서가 이동한다.

여기서는 화살표 키를 이용하여 여러 트랙으로 자유롭게 이동할 수 있다. 이들을 선택/취소하기 위해 [SPACE]키를 사용한다. 선택된 트랙은 '.'이 표시되고, 선택되지 않은 것은 빈칸이 생긴다. 〔+〕키를 입력하면 모든 트랙을 선택하거나 취소한 상태로 전환할 수 있다. 〔−〕키를 누르면 모든 트랙에 대한 정보를 지우게 된다.

뒤에 다시 이야기하겠지만 이미 분석된 트랙은 다시 선택되지 않는다. 이 영역에는 선택된 트랙의 표시에 '.' 대신 다른 문자가 표시되는 경우가 있는데 그 의미는 다음과 같다.

- '–' : 빈트랙
- ',' : 8섹터로 이루어짐
- '.' : 9섹터로 이루어짐
- 'c' : 8이나 9섹터가 아님
- 's' : 첫 섹터의 크기가 다름
- 'i' : 섹터 정보는 올바르지 못함
- 'd' : 섹터 정보가 이중으로 됨
- 'h' : 숨겨짐
- '#' : CRC 에러, 혹은 기타 에러
- 'c' : 비교 결과가 다름 (Compare기능 참조)
- 'v' : 라이트후의 확인 결과가 다름 (Verify)

③ 작업화면 3

이 영역은 선택된 트랙의 데이터 내용을 표시해 준다. 트랙 내용을 변경할 때나 각 기능에 필요한 데이터를 입력할 때 사용한다.

[F3]키를 눌러서 들어온다.

④ 작업화면 4

[F2]키를 누르면 작업화면 4로 이동한다. 여기에는 해당 섹터에 관한 정보가 표시된다. 화면 2의 좌측에 있는 섹터 정보영역을 보자. 맨 윗줄은 선택된 트랙이 몇개의 섹터로 이루어졌는지를 알려 준다. 그리고, 위로부터 2번째에 있는 문자들의 의미는 다음과 같다.

- T : 현재의 섹터가 포함된 트랙
- H : 각 섹터를 읽기 위해 사용된 헤드
- S : 섹터번호
- N : 512 바이트의 자료를 나타낸다.
- % : 각 섹터가 해당 트랙에서 차지하는 양의 백분율
- Stat : 이 값은 디스크 드라이브 콘트롤러로부터 직접 받은 상태값에 약간의 정보가 추가된 것으로 IBM의 테크니컬 레퍼런스를 참고로 하기 바란다.

록스미스에서 사용되는 상태 값들은 다음과 같다.

- 0000 : 읽어낸 후의 상태
- 0020 : CRC 에러
- 0040 : 지워진 데이터 필드
- 0080 : 라이트(Write) 불가
- 0100 : 정보영역에서 번지를 잃어버림
- 0800 : 사용되지 않음
- 2000 : 정보영역의 CRC 에러
- 4000 : CRC가 아직 올바르게 쓰여지지 않음

● 펑션키의 사용

록스미스에는 펑션키를 사용하는 기능이 많지만 별다른 Help기능이 없어서 자세한 사용법을 알기가 어렵다. 다음에 각 펑션키의 기능을 설명해 보겠다. 이 부분 역시 우선 가볍게 훑어 본 후 다음 내용을 읽어 가면서 참고로 하면 도움이 될 것이다.

① **ESC**

록스미스는 모든 기능을 수행하기 전에 이를 다시한번 확인하는데 이 상태에서 ESC키를 눌러 주면 그 기능의 수행을 취소한다. 또한 이 키는 상위 메뉴로 돌아갈 때 사용한다.

② **SPACE**

선택된 기능의 수행을 확인하는 과정에서 SPACE가 눌러지면 기능이 수행된다.

③ **F1**

이 키가 눌러지면 커서는 작업화면 2에 위치한다. 여기서는 처리하고자 하는 트랙을 선택할 수 있다.

④ **F2**

작업화면 4로 커서가 이동한다. 여기서는 섹터정보를 변경할 수 있다.

⑤ **F3**

이 키가 눌러지면 작업화면 3으로 커서가 이동한다. 이 영역에는 선택된 트랙의 데이터가 표시되는데 이들의 값을 직접 변환하거나 각종 기능의 수행에 필요한 문자(숫자)열을 입력할 때 사용한다.

⑥ **F4**

작업화면 3에는 해당 트랙의 내용이 256 바이트만 표시되므로 그 뒤의 내용을 이어서 보고 싶을 경우는 이 키를 눌러준다. 그 이후는 커서 이동키로 아랫쪽 페이지로 이동하여 데이터를 볼 수 있다.

⑦ **F5**

작업화면 4에 반전되어 있는 섹터부터 시작하도록 해 준다.

⑧ **F6**

이 키가 입력된 이후에는 작업화면 3의 영역이 움직인다 하더라도 작업화면 4의 섹터영역의 지시는 바뀌지 않게 된다.

⑨ **F7**

작업화면 3에 나타나는 데이터 값을 비트단위로 오른쪽으로 한자리 이동한 값을 표시한다. 수치적으로는 1/2한 값이 표시된다. 하지만 화면에서의 변화일 뿐 실제의 내용은 바뀌지 않는다.
(예 : 40H→20H→10H→8H…)

⑩ **F8**

F7 키와는 반대로 왼쪽으로 자리이동(Shift)한다.
(예 : 8H→10H→20H…)

① **F9**

이 키가 눌러지면 작업화면3에 나타나는 데이터를 ASCII 문자, 혹은 16진의 숫자로 전환할 수 있다. 문자열을 확인하고자 하는 경우에 매우 편리한 기능이다.

② **F10**

문자열 찾기를 실행한다. 서치(Search) 메뉴를 선택하여 이미 지정한 문자열(숫자)을 찾는다. 그리고 그 결과를 화면에 보여준다.

(2) DOS 프롬프트상에서의 사용

이 방식은 디스크의 복사를 위해 사용된다. 예를 들어 A: 드라이브의 디스크를 B: 드라이브로 복사하는 경우

```
A:>LS A: B:
```

를 입력한다. 이렇게 하면 곧 초기화면과 안내문이 나오는데, 여기서 [SPACE] 키를 한번 눌러주면 록스미스는 많은 복사방지된(Protected) 프로그램을 완전하게 복사해 준다. 하나의 드라이브를 사용하는 경우에는

```
A:>LS A: A:
```

와 같이 하면 된다. 그런데 록스미스를 사용하여 복사를 해보면 PCTOOLS과 같은 프로그램에 비하여 무척 속도가 느린 것을 알 수 있다. 속도가 느린 이유는 록스미스의 경우 프로텍트된 프로그램의 카피를 위한 분석과정을 포함하기 때문이다. 그러므로 프로텍트가 걸리지 않은 디스크를 복사할 때는

```
A:>LS A: B: /S
```

와 같이 S 옵션을 붙여 준다. 그렇게 하면 분석과정을 생략하므로 빠른 속도로 카피할 수 있다.

이 방식의 명령 형식은 다음과 같다.

```
A:>LS [d:][d:][/S][/1]
```

여기서 d:는 원본, 목적 디스크의 드라이브를 지정한다. 다음에는 옵션이 붙는데 앞서 말한 'S'외에 '1'이라는 옵션이 더 있음을 알 수 있다. 이 옵션은 단면 디스크의 복사를 위해 사용된다. 국내의 XT기종에서는 아주 특수한 경우를 제외하고는 디스켓을 단면으로 사용하는 경우가 없을 것이므로, 이 옵션은 거의 사용되지 않는다.

(3) LS 단독으로 사용

DOS상에서 옵션 없이 LS만을 입력하면 LS프로그램이 실행된다. 그리고는 화면 1과 같은 초기 화면이 나올 것이다. 화면의 윗쪽에 있는 메뉴가 LS의 기능들을 표시하는 상위 메뉴이다. 이 메뉴의 첫문자를 키보드로 입력하거나 화살표 키로 이동하여 [RETURN]을 치게 되면 선택이 된다. 선택이 되면 상위 메뉴에 따르는 옵션인 하위 메뉴가 다시 화면 윗줄에 나타난다. 이 상태에서 다시 메뉴를 선택하는 방식으로 원하는 기능을 고르게 된다.

그림 2에 LS.EXE의 모든 메뉴를 나타내었다. 마치 DOS의 서브디렉토리와 같은 트리구조를 하고 있다. 이러한 방식은 많은 프로그램에서 익숙해 있을 것이다. 그럼 이제부터는 LS의 실제 기능을 한가지씩 훑어 보도록 하자.

● **Drive**

이 기능은 LS가 작업할 대상이 되는 디스크를 선택하는 과정이다. LS.EXE는 일반적으로 2 FDD를 가지고 있을 경우 자동으로 A:, B:를 선택해 준다. 한개의 FDD를 가지고 있을 경

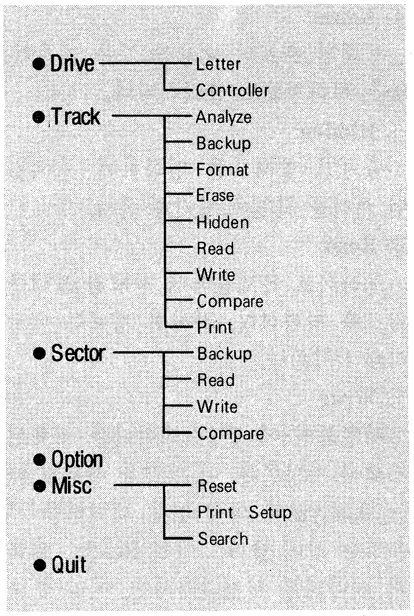
〔화면 1〕 LS. EXE의 초기화면

```
Drive settings                                              Drives: A: » B:
Drive  Track  Sector  Options  Misc  Quit             Locksmith/PC    Ver. 1.3

  Track:   00000000000000001111111111111111222222222    Sectors: 00
           0123456789ABCDEF0123456789ABCDEF0123456789    T  H  S  N  Stat  %
  Head 0:  ..........................................
  Head 1:  ..........................................

[0000] 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
 0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

〔그림 2〕 LS. EXE의 메뉴구조



우나, 소스와 목적 디스크의 순서를 바꾸고자 하는 경우는 DRIVE 메뉴를 선택한 후 다시 LETTER를 선택한다. 그러면 화면에는 소스와, 목적 디스크가 들어갈 드라이브의 철자가 표시되고, 그 자리에 커서가 놓이게 되어 사용자가 변경할 수 있도록 되어 있다. 여기서 원하는 드라이브를 선택한다.

단 앞서 말했듯이 하드디스크는 선택할 수 없다. 또한 디스크 드라이브의 헤드 이동에 까지 관여할 수 있는 CONTROLLER 메뉴는 전문적인 지식이 없다면 함부로 변경하지 않는 것이 좋다.

● Track

① Analyze

선택된 트랙(선택 방법은 일반적인 규칙 참조)을 읽어서 그 트랙의 포맷을 분석한다. 분석된 내용은 작업화면에 나타나게 된다. 우리는 이 화면을 통해 디스켓이 어떤 포맷으로 되어 있는지를 확인할 수 있다. 또한 화면의 좌상단에는 디스크 드라이브 모터의 RPM(분당 회전수)가 표시되며 일단 분석된 트랙에 대한 정보는 메모리에 기억을 하고 있으므로 트랙선택이 되지 않는다(화면 2 참조).

② Backup

선택된 트랙을 읽고, 분석하여(Analyze) 목적 디스크에 복사해 준다. 복사와 함께 올바르게 복사되었는지 비교를 해준다. 만일 원본과 같지 않다면 트랙 영역에 'V'가 표시되는데, 이러한 경우는 다시 복사를 해본다.

③ Format

선택된 트랙은 이미 지정된 섹터정보에 따라 포맷한다. 원하는 트랙만을 포

〔화면 2〕 트랙선택

```
Use arrows to move around inside options                      Drives: A: » B:
Re-type any information to change it, 'ESC' to exit

  Track:   00000000000000001111111111111111222222222     Sectors: 09
           0123456789ABCDEF0123456789ABCDEF0123456789     T  H  S  N  Stat  %
  Head 0:  .....    .......                ..........               02        2.7
  Head 1:  ...      ................       ..........    1 02 01 01 02 0000 10.5
                                                         2 02 01 02 02 0000 10.5
[E9BD] 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F    3 02 01 03 02 0000 10.5
 0000 00 00 00 06 00 00 FF 00 5A 50 01 00 00 00 00 20    4 02 01 04 02 0000 10.5
 0010 00 1F 7F 7F 01 00 00 00 00 00 00 00 00 00 00 00    5 02 01 05 02 0000 10.5
 0020 04 08 12 01 2E C0 27 0A 23 3E 99 CA 47 FF 01 00    6 02 01 06 02 0000 10.5
 0030 02 07 66 25 7F 37 6F 00 00 00 00 00 00 00 00 00    7 02 01 07 02 0000 10.5
 0040 00 00 00 FF FF FF FF 7F 7F 7F 7F 0A 36 40 F0 FF    8 02 01 08 02 0000 10.5
 0050 FF FF FF 08 1A 22 40 C8 C8 C8 00 00 00 00 00 00    9 02 01 09 02 0000 13.5
 0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00D0 00 00 00 66 00 07 00 02 00 00 00 01 00 FF 00 47
 00E0 00 CA 00 00 3E 00 23 00 0A 00 27 6F C0 37 2E 7F
 00F0 C8 00 D8 10 18 20 40 49 4B 4D 47 02 01 FF 01 00
```

맷할 수 있다는 점이 독특하다.

④ **Erase**

선택된 트랙의 내용을 모두 지운다. 물론 섹터정보도 클리어된다.

⑤ **Hidden**

숨겨진 플래그를 변환하여, 숨겨졌던 섹터를 화면에 표시해 준다.

⑥ **Read**

드라이브 콘트롤러에 선택된 트랙을 읽도록 명령한다. 읽혀진 정보는 메모리에 기억된다.

⑦ **Write**

작업화면3에 있는 데이터를 선택된 트랙에 저장한다. 이 명령을 사용할 때는 Analyze에 의해 읽은 데이터를 사용하는 것이 좋다. 만일 Read로 읽혀진 데이터를 저장하려하면, 여러 트랙을 선택한 경우 모두 같은 내용이 저장되게 된다.

⑧ **Compare**

디스크의 내용을 비교하는데 사용된다. 최근에 분석된 디스크 내용의 체크섬과 현재 선택된 트랙 내용의 체크섬을 비교하게 된다. 만일 화면에 'C'자가 표시되면 내용이 다른 것이다.

⑨ **Print**

문자열 혹은 데이터를 프린터로 출력하도록 한다. 이 기능을 선택하면 작업화면3에 커서가 위치하는데, 여기에 프린터로 보낼 데이터를 입력한다. 데이터를 입력하고 [ESC]로 빠져 나오면 프린터로 출력된다.

● **Sector**

① **Backup**

이 복사기능은 트랙 복사에 비해 매우 빠르다. (프로텍트되지 않는 경우에 사용) 복사시에는 자동으로 포맷이 되며, 이미 포맷이 되어 있는 디스크라면 더욱 빠른 속도를 보인다. 선택된 트랙만을 대상으로 작업한다.

② **Read**

디스크의 내용을 메모리로 읽어 들인다. 읽혀진 내용은 다음의 Write, Compare 등에서 사용된다. 읽기를 선택하면 트랙과, 첫섹터, 읽을 섹터의 수, 헤드번호를 물어 온다. 원하는 수치를 입력하면 된다.

③ **Write**

이미 읽혀진 데이터를 디스크에 저장한다. 역시 읽기에서와 마찬가지로 저장범위를 정해준다.

④ **Compare**

읽기에 의해 메모리에 저장되어 있는 내용과 선택된 섹터의 내용을 비교하고 그 결과를 알려 준다.

● **Option**

록스미스가 수행되는데 참고로 하게 되는 지침서라고 할 수 있다. 모두 20H(16진수)개의 옵션번지가 사용되며 각각 고유의 의미가 있다. 기능이 선택되면 화면에는 화면 3이 나타나고, 윗쪽 20H개의 데이터만이 의미가 있다. 여기서 자주 사용되는 몇가지만 언급하도록 하고, 자세한 사항은 매뉴얼을 참조하기 바란다.

- **00**: 이 값이 0이 아니면 디스크의 숨겨진 영역까지 작업 대상으로 한다.
- **0A**: 이 값이 0이 아니면 복사를 할 때 잘 쓰여 졌는지 비교를 한다.
- **0D**: 이 값이 0이 아니면 한번 읽은 데이터를 다시 읽지 않고 여러번 복사할 수 있게 된다. 처음 복사가 끝나면 다시 쓸것인지를 물어 온다. 이때 디스크를 바꿔 넣고 'Y'라고 대답하면 여러장의 복사판을 만들 수 있다.

● **Misc**

①**Reset**

작업영역이나 이미 선택된 트랙, 섹터 등의 선택을 취소한다.(초기화)

② **Print Setup**

TRACK에서 Print 내용을 참조하기 바란다.

③ **Search**

디스크의 내용 중 원하는 데이터를 찾을 수 있는 기능이다. 이 기능을 선택하면 작업화면3에 커서가 위치한다. 처음부터 찾고자 하는 데이터를 입력한다. 만일 문자로 입력하고자 하는 경우에는 F9키를 눌러서 문자모드와 16진 모드를 번갈아 사용할 수 있다. 원하는 데이터의 입력이 끝났으면 [ESC]키를 눌러서 빠져 나오면 찾을 데이터의 길이를 물어온다. 여기서 16진수로 입력한다. 그 다음에 [F10]키를 누르면 데이터를 찾을 수 있다.

〔화면 3〕 옵션변경

```
Use arrows to change tracks and heads, 'SPACE' to toggle      Drives: A: » B:
Press '+' to toggle all, '-' to clear tables, ESC to exit

  Track:   00000000000000001111111111111111222222222   |     Sectors: 09
           0123456789ABCDEF0123456789ABCDEF0123456789  |   T  H  S  N  Stat  %
  Head 0:  .....    .......             ..........     |            02        2.7
  Head 1:  ...      ................    ..........     | 1 02 01 01 02 0000 10.5
                                                       | 2 02 01 02 02 0000 10.5
[E9BD] 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  | 3 02 01 03 02 0000 10.5
 0000 DB 32 FF 88 3E 03 03 C6 06 04 03 01 88 3E F8 02  | 4 02 01 04 02 0000 10.5
 0010 88 3E F6 02 8A DC D1 E3 FC 0A E4 74 16 80 FC 59  | 5 02 01 05 02 0000 10.5
 0020 74 3D 80 FC 0C 77 0C 80 3E CE 02 00 75 31 BC 06  | 6 02 01 06 02 0000 10.5
 0030 0A EB 2C A3 E6 02 C6 06 D1 02 01 C6 06 CE 02 00  | 7 02 01 07 02 0000 10.5
 0040 C6 06 D0 02 FF 50 B4 82 CD 2A 58 C6 06 04 03 00  | 8 02 01 08 02 0000 10.5
 0050 BC 86 08 F6 06 E5 02 FF 74 05 50 E8 B1 3B 58 2E  | 9 02 01 09 02 0000 13.5
 0060 8B 9F EF 0D 87 1E 7C 05 8E 1E 7E 05 36 FF 16 7C  |
 0070 05 FA 2E FE 0E CF 02 2E 8E 16 20 05 2E 8B 26 1E  |
 0080 05 8B EC 88 46 00 2E A1 84 05 2E A3 1E 05 2E A1  |
 0090 82 05 2E A3 20 05 E8 01 00 CF 2E 8F 06 80 05 58  |
 00A0 5B 59 5A 5E 5F 5D 1F 07 2E FF 26 80 05 2E 8F 06  |
 00B0 80 05 06 1E 55 57 56 52 51 53 50 2E FF 26 80 05  |
 00C0 2E C5 36 1E 05 C3 26 81 4F 03 00 01 CB 36 FE 06  |
 00D0 CF 02 FB FC 1E 16 1F E8 92 00 1F 73 05 B4 02 B0  |
 00E0 01 C3 26 C7 46 1E FF FF 1E 56 50 16 1F BE 60 03  |
 00F0 88 04 80 04 41 C7 44 01 3A 00 B8 00 03 F8 CD 2A  |
```

● **Quit**

LS.EXE의 사용을 종료하고 DOS로 돌아간다.

LSF.EXE(화일버전)

이 프로그램은 룩스미스의 화일용 버전이다. 따라서 LS.EXE가 디스크를 대상으로 작업하는 반면 LSF.EXE는 화일단위의 작업을 할 수 있는 것으로 일반적인 디스크 매니저 프로그램이 갖는 기능은 대부분 갖고 있다. 사용방법이나 기능이 다른 프로그램과 비슷하므로 간단히 설명하도록 하겠다.

메뉴 구조는 **그림 3**와 같다. 일반적인 사용방법은 LS.EXE에서와 같다. 그러면 이제 메뉴별로 기능을 알아보자.

● **Drive**

이 기능은 LS.EXE에서와 같이 대상 드라이브를 정해주는 것이다. 단 이 프로그램은 화일을 대상으로 작업하기 때문에 하드디스크도 사용가능하다.

● **File**

① **Directory**

현재의 디렉토리에 있는 화일들을 작업화면4에 보여준다. 작업화면3에는 각 화일들이 디스크에 어떤 모습으로 저장되어 있는지에 대한 지도가 나타난다(**화면 4** 참조). 여기서 F2를 눌러서 작업화면4로 들어가면 커서 이동키를 사용하여 원하는 화일을 선택하여 그 화일이 디스크에 배치된 위치를 확인 할 수 있다.

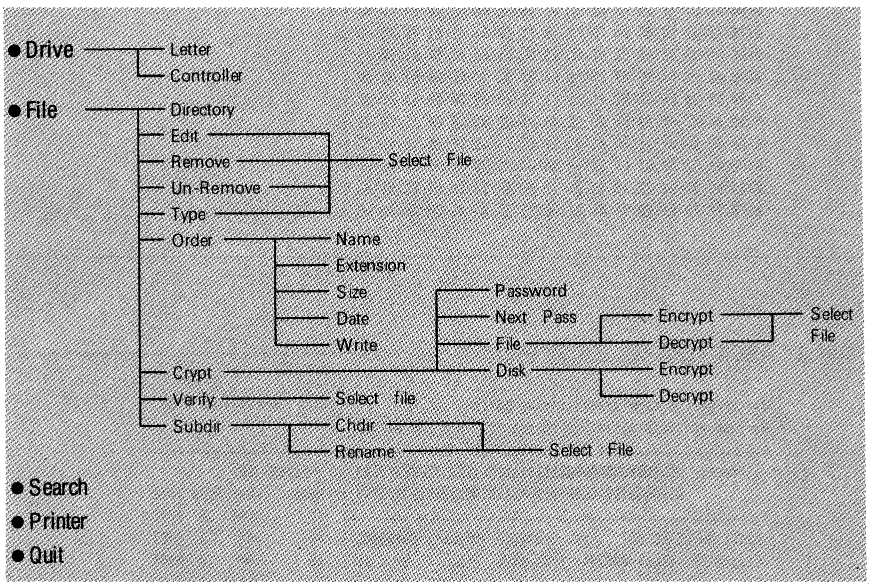
② **Edit**

이 기능은 화일의 내용을 변경할 수 있는 기능이다. PCTOOLS의 Edit 기능과 같은 목적으로 사용된다. Edit를 선택한 후 커서 이동키로 원하는 화일을 고르면 작업화면3에는 그 화일의 내용이 나타난다(**화면 5** 참조). 여기서 직접 키보드로 원하는 부분을 고치면 된다. [F9]를 누르면 표시되는 데이터가 ASCII문자로 바뀌고, 다시 누르면 환원된다(**화면 6** 참조). 다 고친 후에는 [ESC]를 누른다. 그러면 작업화면1에는 정말 고칠 것인지를 확인하는 안내문이 나온다. 여기서 [SPACE]를 누르면 고친 내용으로 저장되고, [ESC]를 누르면 고치기 전의 내용으로 환원되어 다시 원래의 상태로 되돌아간다.

③ **Remove**

화일을 지우는 기능이다. DOS의 DEL과 같은 기능을 수행한다. 사용법은 일반적인 규칙을 따른다. 일단 지워진 후에는 변경 후의 디렉토리를 다시 보여주고 계속 작업을 할 수 있도록 만들어 준다.

〔그림 3〕 LSF.EXE의 메뉴구조



〔화면 4〕 LSF.EXE의 Directory기능

```
Get root directory from disk                        Drives: B: » B:
Directory  Edit  Remove  Un-remove  Type  Order  Crypt  Verify  Subdir
─────────────── Dos 2.0 Double sided ──────────────────────────────────
   Track:  000000000000000011111111111111112222222222 │  Files: 09
           0123456789ABCDEF0123456789ABCDEF0123456789 │ Name    Ext  Type Size
──────────────────── Head 0 ──────────────────────────│ IO      .SYS  27  0020
   Sec 1:  B##########################...##.......... │ MSDOS   .SYS  27  0038
   Sec 2:  F##########################...##.......... │ COMMAND .COM  20  0030
   Sec 3:  F##########################..###.......... │ LS      .EXE  20  0092
   Sec 4:  F##########################..###.......... │ LSF     .EXE  20  0066
   Sec 5:  F##########################..###.......... │ LSK1    .EXE  20  0004
   Sec 6:  D##########################..###.......... │ LS      .BAK  20  002A
   Sec 7:  D##########################..###.......... │ MAP     .     20  0002
   Sec 8:  D##########################..###.......... │ LS      .GEM  20  002C
   Sec 9:  D##########################..###.......... │
──────────────────── Head 1 ──────────────────────────│
   Sec 1:  D##########################..###.......... │
   Sec 2:  D##########################..##........... │
   Sec 3:  D##########################..##........... │
   Sec 4:  ##########################...##........... │
   Sec 5:  ##########################...##........... │
   Sec 6:  ##########################...##........... │
   Sec 7:  ##########################...##........... │
   Sec 8:  ##########################...##........... │
   Sec 9:  ##########################...##........... │
```

〔화면 5〕 16진 데이터로 표시된 편집 화면

```
Use arrows to move around inside buffer                       Drives: B: » B:
Re-type any information to change it, 'ESC' to exit
──────────────── Dos 2.0 Double sided ──────────────────────────────────────
  Track:  0000000000000000111111111111111122222222222 │  Files: 09
          0123456789ABCDEF0123456789ABCDEF0123456789 │ Name     Ext  Type Size
─────────────────────────────────────────────────────│ IO       .SYS  27  0020
 File: LS      .GEM  Size: 00005485  Offset: 00000100 │ MSDOS    .SYS  27  0038
 Clu: 00F0  Lsec: 0001E8  Trk: 001B  Head: 0  Sec: 03 │ COMMAND  .COM  20  0030
─────────────────────────────────────────────────────│ LS       .EXE  20  0092
 0100 61 9F 69 20 88 61 93 77 D0 61 89 41 20 D0 81 20 │ LSF      .EXE  20  0066
 0110 BA 81 A1 61 2C 20 97 A1 AF 61 C7 61 B7 81 20 90 │ LSK1     .EXE  20  0004
 0120 81 B6 77 B7 69 0D 0A BB A2 B8 F3 20 A5 65 89 77 │ LS       .BAK  20  002A
 0130 D0 69 20 AE 81 20 B7 B6 95 A1 9D A2 20 D0 81 BA │ MAP      .     20  0002
 0140 85 94 61 2E 20 99 A1 D0 65 20 50 43 54 4F 4F 4C │ LS       .GEM  20  002C
 0150 53 97 77 B7 A1 20 88 61 BB A1 93 65 20 97 A1 AF │
 0160 61 C7 61 20 A0 81 93 A1 B8 E1 B7 81 20 8B A1 93 │
 0170 77 B7 69 0D 0A CD A1 D0 71 D0 61 A1 61 2C 20 88 │
 0180 81 B7 A5 B8 E2 B7 A5 20 CC 61 B7 A9 B7 81 20 A5 │
 0190 A1 D1 A1 9F 69 20 B6 E1 D0 81 20 CC 61 B7 A9 B7 │
 01A0 69 20 B4 71 D1 A1 D1 C1 20 D0 69 20 AE 81 20 B7 │
 01B0 B6 B7 61 A1 61 2C 20 94 61 9F 65 20 C4 61 CF A1 │
 01C0 0D 0A CF 61 9D A1 8B 61 9C 91 89 C1 20 94 69 9F │
 01D0 A1 20 D0 65 A4 E5 B7 AA B7 65 20 97 A1 AF 61 C7 │
 01E0 61 B7 81 20 90 81 B6 77 B7 69 20 B5 61 9C E1 B8 │
 01F0 77 B7 81 20 97 A1 AF 61 C7 61 B5 41 20 A4 65 A5 │
```

〔화면 6〕 ASCII 캐릭터로 표시된 편집화면

```
Use arrows to move around inside buffer                       Drives: B: » B:
Re-type any information to change it, 'ESC' to exit
──────────────── Dos 2.0 Double sided ──────────────────────────────────────
  Track:  0000000000000000111111111111111122222222222 │  Files: 1A
          0123456789ABCDEF0123456789ABCDEF0123456789 │ Name     Ext  Type Size
─────────────────────────────────────────────────────│ TP       .BAT  20  0002
 File: AUTOEXEC.BAT  Size: 000000C2  Offset: 00000000 │ DB       .BAT  20  0002
 Clu: 013E  Lsec: 000284  Trk: 0023  Head: 1  Sec: 06 │ DS       .BAT  20  0002
─────────────────────────────────────────────────────│ VL       .BAT  20  0002
 0000 e  c  h  o     o  f  f  .  .  p  r  o  m  p  t  │ LL       .BAT  20  0002
 0010    $  p  $  g  .  .  p  a  t  h     c  :  \  d  │ NC       .BAT  20  0002
 0020 o  s  ;  c  :  \  s  e  r  \  b  a  t  ;  c  :  │ NCD      .BAT  20  0002
 0030 \  s  e  r  \  s  c  r  u  t  i  ;  c  :  \  s  │ PC4      .BAT  20  0002
 0040 e  r  \  p  r  t  u  t  i  ;  c  :  \  s  e  r  │ GEM      .BAT  20  0002
 0050 \  c  o  m  m  u  t  i  ;  c  :  \  s  e  r  \  │ DIS      .BAT  20  0002
 0060 d  i  s  k  u  t  i  ;  c  :  \  s  e  r  \  h  │ REV      .BAT  20  0002
 0070 e  l  p  u  t  i  ;  c  :  \  s  e  r  \  m  o  │ TALK     .BAT  20  0002
 0080 u  s  e  u  t  i  .  .  t  i  m  e  r  /  s  .  │ MEDI     .BAT  20  0002
 0090 .  \  u  t  i  l  i  t  y  \  p  c  t  o  o  l  │ MASM     .BAT  20  0002
 00A0 s  \  p  c  -  c  a  c  h  e     /  i  a  /  i  │ MOVE     .BAT  20  0002
 00B0 b  .  .  i  n  s  t  a  l  l  .  .  c  l  s  .  │ REMOVE   .BAT  20  0002
 00C0 .  .  S  K  U  T  I                 .  .  .  .  .  │ TETRIS   .BAT  20  0002
 00D0 .  .  .  .  .  .  7  .  !  .  I  .  .  .  .  .  │ FILEPRNT.BAT  20  0002
 00E0 H  E  L  P  U  T  I                 .  .  .  .  .  │ AUTO     .BAT  20  0002
 00F0 .  .  .  .  .  .  :  .  !  .  J  .  .  .  .  .  │ AUTOEXEC.BAT  20  0002
```

④ **Un-remove**

지워진 화일을 복구시켜 주는 기능이다. 보통 DEL로 지워진 화일은 화일명의 첫자가 바뀌기 때문에 이 기능을 선택하면 지워진 화일들이 작업화면4에 나타난다. 이들 화일들은 모두 첫글자가 바뀌어 있을 것이다. 커서키로 원하는 위치에 가서 그 화일의 첫 글자를 키보드로 입력해 주면 그화일은 복구된다.

하지만 이미 그 화일이 있던 위치에 다른 화일이 겹쳐지거나 손상되었을 경우는 복구할 수 없다. 그러므로 실수로 화일을 지웠을 경우는 절대로 그 디스크에 데이터를 저장하지 말고, 일단 복구를 시켜야 한다.

⑤ **Type**

이 기능은 화일의 속성(Attribute)을 변경하도록 해준다. 그러니까 이 기능을 사용하면 화일을 숨기거나 읽기만 가능하도록 할 수도 있다. 이 기능을 선택하면 작업화면4에, 화일명과 숫자가 나오는데 이 숫자가 그 화일의 속성을 알려준다. 그 숫자의 의미는 다음과 같다.

- 01 : 읽는 것만 가능
- 02 : 이 화일은 숨겨져 있다. dir로는 볼 수 없음
- 04 : 시스템 화일(.sys)
- 08 : 디스크의 볼륨명
- 10 : 서브 디렉토리
- 20 : 닫혀진 클러스터이다.

원하는 위치에 가서 키보드로 변경할 내용을 입력하면 화일의 속성을 변경할 수 있다.

⑥ **Order**

이 기능은 디렉토리의 정렬(Sort)을 위해 사용된다. 화일명, 확장자별, 화일 크기순, 생성 날짜순으로 정렬할 수 있다. 정렬 후에는 반드시 Write 메뉴를 선택하여 저장을 하여야 한다. 그렇지 않으면 변경되지 않는다.

⑦ **Crypt**

중요한 데이터나 개인적인 내용을 다른 사람이 볼 수 없도록 하고 싶을 때가 있을 것이다. 이 기능은 화일을 암호화하여 다른 사람이 사용할 수 없도록 해준다. 우선 선택을 하면 패스워드(Password)를 물어온다. 8자 이내로 대답을 하면 다시한번 확인을 한다. 다음에 File을 선택하여 Encrypt를 선택하고, 원하는 화일위치에서 [RETURN]을 치면 그 화일은 암호화된다. 다시 복구를 하고자 할 때는 Decrypt를 사용하면 된다. 만일 패스워드를 잊은 경우는 복구가 어려우므로 조심하기 바란다.

File 대신 Disk를 선택하면 디스크 전체를 암호화 할 수 있다.

⑧ Verify

이 기능을 선택하면 선택된 화일의 내용을 모두 읽어서 그 체크섬을 알려준다. 만일 같은 내용을 가진 화일이라면 같은 체크섬을 가질 것이므로 화일의 비교를 할 때 사용된다.

⑨ Subdir

디스크에는 다수의 서브디렉토리가 있는 경우가 많다. 그러므로 작업영역을 서브 디렉토리로 옮길 수 있도록 해준다. 그리고 화일의 이름을 다른 것으로 바꾸어 줄 수도 있다.

●Search

디스크의 내용 중 원하는 데이터를 찾을 수 있는 기능이다. 이 기능은 LS.EXE에서와 같은 방식으로 사용하면 된다.

●Printer

이 기능은 문자열 혹은 데이터를 프린터로 출력할 수 있도록 할 수 있다. 역시 LS.EXE에서의 기능과 같은 방식으로 사용한다.

●Quit

LSF.EXE의 수행을 종료하고 DOS로 돌아간다.

마치며

지금까지 알아본 바와 같이 록스미스는 디스크의 세밀한 부분까지 들여다 볼 수 있도록 해 준다. 우리 주변에는 유용한 유틸리티들이 많이 있지만 그중의 많은 프로그램들이 무단 복제에 의해 배포되고 있기 때문에 프로그램의 입수와 함께 매뉴얼과 같은 지침서를 접할 기회가 그리 많지 않아서, 많은 사람들이 주위의 사람들로부터 들은 지식이나, 시행착오의 과정을 통해 사용법을 익히게 된다. 이 방법은 매우 위험한 것이다. 디스크를 직접 관리하는 록스미스와 같은 프로그램을 자칫 잘못 사용하면 귀중한 데이터나 프로그램을 잃게 되는 경우가 있다. 가능한한 모든 프로그램은 매뉴얼을 찾아 읽어 보는 것을 습관화하기 바란다. 매뉴얼을 구하기가 쉽지 않은 경우라 하더라도 국내의 여러 잡지들을 통해 대부분의 유용한 프로그램들에 대한 자료를 찾아 볼 수 있을 것이다. 이러한 태도가 습관화되면 여러 프로그램들의 아까운 기능들을 그냥 지나치는 일은 없게 될 것이다. 특히 프로그래밍에 관심이 있는 사람에게는 상업화된 프로그램들의 구조와 독특한 기능들은 좋은 참고자료가 된다. PC

〈용어해설〉

●체크섬(check sum)
합산검사에서 사용하는 합계.

●합산검사(check summation)
특정한 그룹안에 있는 수들을 더해서 그 합계를 이미 계산되어 있는 검사 합계와 비교하여 자료의 정확성을 통제하는 검사임.

●정렬(sort)
정보의 레코드를 키항목이나 필드값에 따라 일정하게 배열하는 것. 또는 각 화일에 포함된 키에 따라 순서대로 화일을 배열하는 시스템 프로그램.

●패스워드(password)
시스템을 보안하도록 화일 또는 시스템에 준 이름 또는 문자의 집합. 화일 또는 시스템을 액세스하기 전에 사용자는 암호를 사용하여야 함.

●볼륨(volume)
어떤 기억 매체로 한 개의 판독/기록 장치로서 접근할 수 있는 부분. 디스크 팩, 자기테이프의 릴과 같이 그 자체를 한 단위의 장치로서 시스템에 연결하거나 분리시킬 수 있는 기억 매체.

●속성(attribute)
어떤 개체(entity)의 성질을 세분화하여 묘사한 것, 하나하나의 개체에 대한 정보가 유지되며, 속성이란 이들 개체가 갖는 관심의 대상이 되는 성질을 말함.

●화일 분석(analyze file)
화일의 특성을 조사, 분석하여 유사성과 중복된 양 및 형태를 결정하고 화일내의 자료 구성 요소에 영향을 주는 문서에 레이블을 부여하거나 리스트 작성을 검사하는 것.

●제어기(controller)
피제어 변수나 시스템을 제어하기 위하여 자동적으로 작동하는 모듈 혹은 장치의 부분.

●검증(verify)
사람이 작성한 자료 재생에서 오류를 줄이기 위하여 자동 검사기 등을 이용하여 비교 검사하는 것. 컴퓨터에 입력할 자료를 만들 때 준비된 자료가 정확하다는 것을 확인하는 것.

●시스템 호환성(compatibility systems)
복잡한 시스템 응용에서 다른 시스템 부분 사이에 전기적, 논리적, 기계적으로 완전한 호환성을 지닌 모듈이 있는데 그 예로는 디지탈 컴퓨터, 입출력 장치들 및 아날로그 접속 장치에서 그 예를 찾아 볼 수 있음.

●화일 보호(file protection)
자료의 우발적 삭제를 보호하는 방법이나 장치를 말함.

●접근시간(access time)
어떤 데이터가 불리어지는 순간과 이 데이터가 전달되는 순간 사이의 시간 간격.

●원소(cell)
주소에 의해 지정되며 기억장치로서의 기능을 갖는 한 위치. 경우에 따라서는 행, 필드, 위치 혹은 블럭이라고 표시되기도 한다.

●가상 시스템(virtual system)
사용자가 시스템을 사용할 때 그 시스템이 실제로 가지고 있는 것보다도 더 많은 주기억 장치를 가지고 있는 것처럼 사용할 수 있도록 하는 운영 체제의 한 형태.

컴퓨터 기본상식 2
직렬통신이란?

조길재/(주)다우기술 개발부 과장

이번호에서는 우리가 컴퓨터를 이용하여 다른 컴퓨터나 다른 사용자, 혹은 보조장치 등과 데이터를 주고 받게하는 직렬통신 어댑터와 이와 관련된 기본상식과 동작원리, 또 그것과 연결하여 사용할 수 있는 장치나 응용에 대하여 알아보기로 하자. 직렬통신에 대한 상식이나 지식이 없는 초보자를 위한 난이므로 부담없이 읽기 바란다.

우리는 주위에서 통신이라는 단어를 흔히 접하게 된다. 통신이라는 단어 자체가 워낙 광범위한 의미를 내포하고 있지만 간단히 말하자면 데이터(자료) 혹은 정보를 주고 받는 행위라고 할 수 있을 것이다(데이터나 정보는 그 의미가 다르나 본고의 이해나 표현의 편의상 앞으로는 데이터로 통일하여 쓰기로 한다).

그러면 어떻게 데이터를 주고 받는가 하는 질문에 이르게 된다. 우선 컴퓨터를 이용하여 데이터를 주고 받고자 할 경우 크게 두가지를 생각해 볼 수 있다. 첫째는 데이터가 들어있는 매체(Media)를 직접 주고 받는 방법이다. 예를 들면 갑이 사용하는 컴퓨터에 들어 있는 데이터를 플로피 디스크나 테이프 등의 매체에 기록한 다음 이 매체를 을에게 전달하고 을 또한 이러한 방법으로 갑에게 매체를 전달한다.

이러한 방법은 갑과 을이 가까운 거리에 있을 경우에는 가능한 통신방법이기는 하나, 매체를 직접 전달하여야 하므로 번거롭기 그지없는 방법이다.

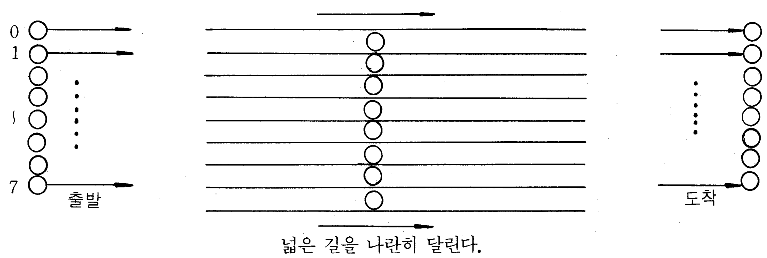
둘째로는, 갑과 을이 사용하는 컴퓨터를 물리적인 케이블로 연결하는 방법이다. 이렇게 연결이 된다면 갑과 을은 자리를 이동하지 않고서도 서로 데이터를 주고 받을 수 있게 된다. 이러한 방법은 오늘날 널리 사용되는 방법으로, 서로가 케이블로 연결이 되어 있으면 손쉽게 데이터를 주고 받을 수 있다.

이 방법은 또 케이블이 어떻게 연결되는가에 따라 **그림 1**과 같이 병렬(Parallel) 통신과 직렬(Serial) 통신으로 나뉘어진다. 쉽게 말하자면, 병렬통신이란 갑과 을이 데이터를 주고 받을 때, 하나의 데이터를 표현하는 최소단위인 바이트(1바이트는 8비트로 구성된다) 이상의 단위로 데이터를 주고 받는 방법이며, 직렬통신이란 바이트의 데이터를 다시 비트로 분리하여 이 비트들을 순차적으로 주고 받는 방법이다.

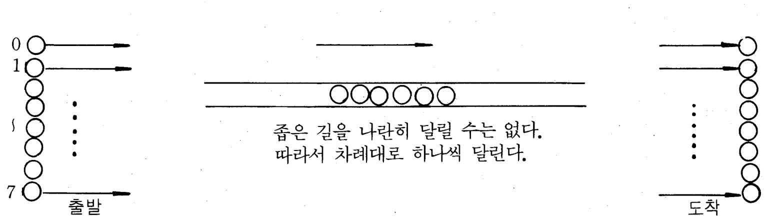
이러한 통신방법은 또 두 컴퓨터를 연결하는 케이블의 종류에 의하여 결정이 된다. 즉 병렬통신의 경우, 하나의 데이터를 전달하려면 데이터의 최소단위인 1바이트, 즉 8비트의 데이터를 한꺼번에 전송 또는 수신할 수 있는 케이블이 필요하다. 말하자면 데이터를 전달하는데 필요한 케이블의 가닥 수가 적어도 8가닥 이상이 필요한 셈이다. 이의 예로는 컴퓨터와 그의 주변장치인 프린터를 연결하는 케이블을 들 수 있다. 실제로 둘을 연결하는 케이블은 18가닥 이상의 케이블로 구성된다.

반면 직렬통신의 경우에는 비트를

〔그림 1〕 병렬통신과 직렬통신



(a) 병렬통신



(b) 직렬통신

주고 받을 수 있는 최소 2가닥의 케이블만 있으면 송수신이 가능하다. 독자 여러분이 흔히 들어서 아는 RS-232C라고 하는 통신방식은 최소한 3가닥이면 두 컴퓨터간의 교신이 가능하게 되어 있는 것이다. 통신 속도면에서 보면 병렬통신방식이 직렬통신방식보다 데이터의 전송속도가 훨씬 빠르나, 경제적인 측면에서 보면 병렬통신방식보다 상대적으로 케이블의 가닥 수가 적게 필요하고 그 구성에서도 간단한 직렬통신방식이 훨씬 유리하며 전화선과 같은 일반 회선을 사용할 수 있는 잇점이 있다.

이제부터는 이 직렬통신—그 중에서도 비동기식 통신—이 어떻게 이루어지는가에 대하여 IBM-PC 호환기종용 비동기식 통신 어댑터를 위주로 구체적으로 알아보기로 한다.

바이트에서 비트로 비트에서 바이트로

독자 여러분은 8비트 컴퓨터 혹은 16비트 컴퓨터라는 말을 자주들었을 것이다. 이 개념은 중앙처리장치(CPU)의 데이터처리 단위의 크기가 얼마인가에 따라 분류한 것으로 요즈음 가장 널리 보급되어 있는 IBM-PC XT 및 AT 호환기종은 16비트 컴퓨터에 속한다. 물론 이들 중앙처리장치들은 8비트의 데이터도 취급할 수 있다.

인텔(INTEL)사의 중앙처리장치인 8088은 외부적으로는 8비트의 데이터 버스를 가지고 있지만 그 내부적으로는 16비트 데이터의 처리를 하므로 16비트 컴퓨터라고 할 수 있다. 컴퓨터 내의 모든 데이터는 이와 같이 8비트 혹은 16비트의 단위로 데이터가 병렬 처리되는데 그렇다면 이들 병렬 데이터를 가지고 다른 컴퓨터와 어떻게 직렬통신을 할까 하는 의문점이 생긴다.

이에 대한 대답은 비동기식 통신 어댑터(Asynchronous Communication Adapter)사의 8250이라고 하는 ACE (Asynchronous Communications Element) LSI가 가지고 있다. 이 LSI의 역할은 그림 2에서 볼 수 있듯이 중앙처리장치로부터 병렬 데이터를 받아서 이 데이터를 직렬 데이터로 변환하고, 또 케이블을 통하여 받아들인 직렬 데이터를 병렬 데이터로 변환하여 중앙처리장치로 보내는 역할을 한다.

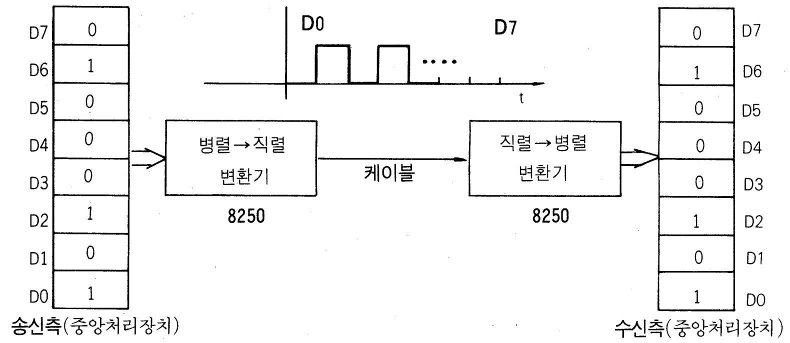
뿐만 아니라 이 LSI는 모뎀으로부터의 입력 신호처리 및 모뎀을 제어하는데 필요한 신호를 발생하는 능력을 가지고 있다. 그림 3에 비동기식 통신 어댑터의 블록 다이어그램을 보였다. 여기에서 한가지 언급하여야 할 것은 주파수 발생기의 역할이다. 후에 상세히 언급되겠지만 이 발생기에서 나오는 클럭이 다른 컴퓨터와의 통신속도를 결정하는 주 요소가 된다는 점이다.

비동기식 통신의 통신 매개변수

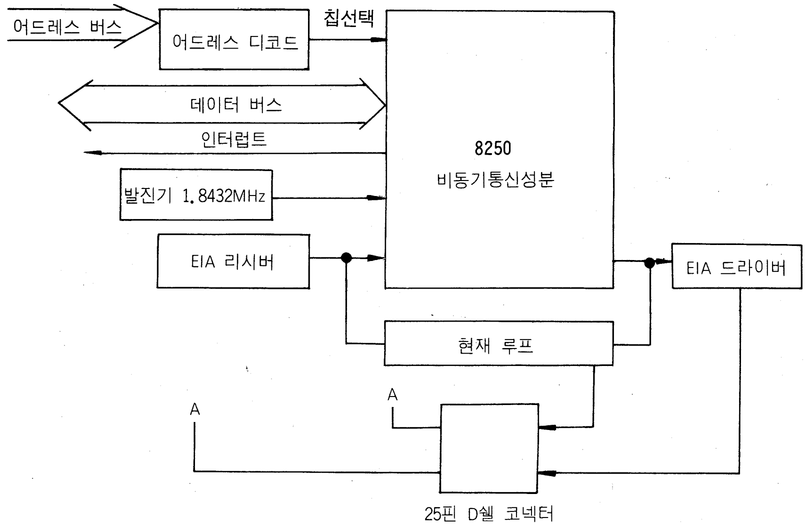
두대의 컴퓨터가 상호 교신을 하기 위해서는 미리 규정된 약속을 지켜야 할 것임에는 틀림 없다. 이것은 마치 언어와 그 언어에 맞는 문법과 같아서 그 언어와 문법에 따라 사용하지 않으면 상호 의사소통이 불가능한 것과 같다.

비동기식 통신에서는 이런 언어와 같은 규정이 있어서 이 규정을 지켜야만 통신이 가능하게 되는 것이다. 이들

(그림 2) 병렬→직렬, 직렬→병렬 데이터변환



(그림 3) 비동기식 통신 어댑터 블록 다이어그램



중 가장 중요한 매개 변수로는 다음과 같은 것들이 있다.

① 통신속도(Baud Rate)
② 데이터길이(Word Length)
③ 패리티(Parity)
④ 스톱비트(Stop bits)

그러면 이들에 대해 살펴보자.

●통신속도

통신속도는 시간당 데이터를 송수신할 수 있는 양을 나타내는 척도이다. 통신속도를 이야기할 때 흔히 BPS라는 약어를 접하게 되는데 이는 Bits Per Second의 약어로 글자 그대로 1초당 전송가능한 비트 수를 의미한다. 비동기식의 통신속도로는 50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200 BPS의 통신속도가 사용되나, PC용 비동기식 어댑터에서는 이들 중 110, 150, 300, 600, 1200, 2400, 4800 및 9600BPS만을 사용한다.

●데이터길이

하나의 데이터를 전송하는데 필요한 데이터의 길이, 즉 비트 수를 나타내는데 이의 종류로는 5, 6, 7, 8 등 네가지가 있다. 7비트 ASCII의 데이터는 데이터의 길이를 7로 하면 가능하나, 표준 한글코드의 경우 2개의 8비트를 사용하므로 이 경우에는 데이터의 길이를 8로 설정하여야 한다.

●패리티

데이터통신은 케이블을 통하여 이루어지는데, 선로상을 통하여 데이터가 오가는 동안에 외적인 요인에 의하여 신호의 흐름이 방해를 받아 데이터가 상대방에게 정확하게 전달되지 않을 수가 있다. 패리티라 함은 이런 방해에 의하여 잘못 수신된 데이터를 발견하기 위해 고안된 것으로, 데이터를 전송할 때 이 데이터에 패리티비트라는 것을 추가하여 데이터의 수신상태를 검사할 수 있도록 한 것이다.

이러한 패리티비트를 사용할 경우에는 짝수방식 및 홀수방식이 있으며 패리티비트를 사용하지 않을 경우도 있다. 예를 들면 영어 대문자 'E(01000101)'를 전송할 때 짝수 패리티를 사용하는 경우 'E'의 데이터열에는 1에 해당하는 비트수가 3개, 즉 홀수개이므로 패리티비트를 1로 하여 전체 1에 해당하는 비트 수를 짝수개로 하며(010001011), 홀수패리티를 사용할 경우에는 패리티비트를 0으로 하여 전체의 1에 해당하는 비트의 수를 홀수로 맞추게 된다(010001010).

여기서 두가지 유의할 것은, 데이터 길이가 8일 경우에는 보통 패리티 비트를 사용하지 않으며, 또한 패리티 비트를 사용한다 하더라도 에러 비트의 수가 짝수이면 에러검증이 되지 않는다는 사실이다.

●스톱비트

비동기식 통신에서는 데이터의 송수신이 비동기적으로 일어나기 때문에, 데이터의 시작과 끝을 알리는 스타트비트와 스톱비트를 사용하고 있다. 이들 시작과 끝을 알리는 비트는 로직레벨이 1로 규정되어 있으며 스타트비트는 1비트의 데이터길이를 가지며, 스톱비트는 1과 1.5, 2비트 등 세가지 종류가 있다.

이와 같은 통신매개 변수의 사용 예로 영문 데이터 'E'를 들어 **그림 4**에 나타내었다.

8250 LSI의 라인 상태레지스터

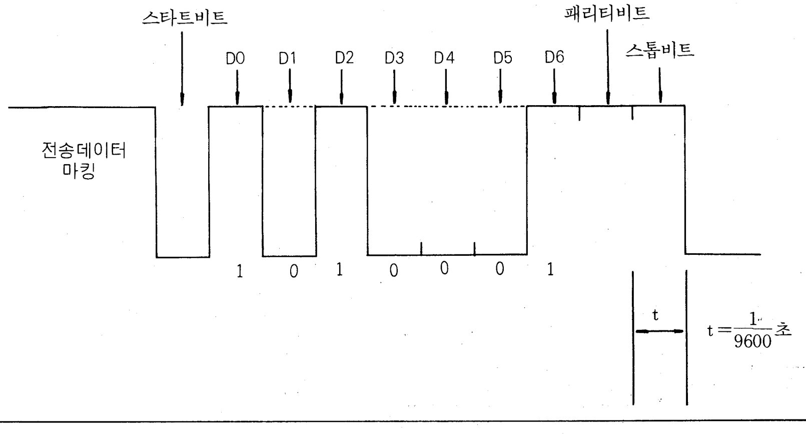
8250 LSI내에는 데이터의 송수신 상태를 나타내는 라인 상태레지스터가 있다.

이 상태레지스터의 각 비트의 내용을 짚고 넘어갈 필요가 있다. 왜냐하면 비동기식 통신을 지원하는 LSI의 대부분이 이와 대동소이한 상태레지스터 혹은 출력포트를 가지고 있기 때문에 차후 이와 관련된 내용이 나오더라도 독자 여러분은 자신을 가지고 이러한 용어들을 대할 수 있을 것이다.

라인 상태레지스터의 내용은 다음과 같다.

① 비트 7 : =0
② 비트 6 : 송신 시프트레지스터 준비완료(TX Shift Register Ready)
③ 비트 5 : 송신 데이터레지스터 빔(Transmiter Holding Register Empty)
④ 비트 4 : 브레이크 인터럽트(Break Interrupt)
⑤ 비트 3 : 프레임 에러(Framing Error)
⑥ 비트 2 : 패리티 에러(Parity Error)
⑦ 비트 1 : 오버런 에러(Overrun Error)
⑧ 비트 0 : 수신 데이터 준비(Data Ready)

〔그림 4〕 직렬전송 데이터(9600BPS, 7비트, 짝수 패리티, 1 스톱비트)



• 비트 6

병렬 데이터를 직렬 데이터로 변환하여 데이터의 송신이 완료되면 이 비트가 세트된다. 즉 데이터의 전송이 끝났음을 알려준다.

• 비트 5

중앙처리장치에서 8250 LSI로 송신데이터를 보내면 8250 LSI는 이 데이터를 송신데이터 레지스터(Transmit Holding Register)에 일단 저장한 후 이 데이터를 송신 시프트레지스터로 보내어 직렬송신을 하게 된다. 이때 상태비트는 이 송신 데이터 레지스터의 상태를 나타내는 것으로 그 값이 1이면 송신 데이터 레지스터가 비어 있다는 것이다. 말하자면 8250LSI가 중앙처리장치로부터 데이터를 받을 준비가 되어 있다는 것을 나타낸다.

• 비트 4

브레이크신호(한 바이트를 전송하는데 걸리는 시간보다 긴 로직레벨 0이 계속되는 경우)를 받으면 이 비트가 세트된다.

• 비트 3

데이터 수신 중 마지막 비트 뒤에 따르는 스톱비트의 로직레벨이 0일 때 상태레지스터의 이 비트가 세트된다.

• 비트 2

수신된 데이터가 규정하고 있는 패리티와 일치하지 않을 때 이 비트가 세트된다.

• 비트 1

이미 수신된 데이터를 중앙처리장치에서 미처 읽어 가기전에 다음의 데이터가 도착하게 되는 경우 이전에 수신된 데이터의 내용은 사라지게 된다. 이 경우를 데이터 오버런이라고 하는데 상태레지스터의 이 비트는 이러한 직렬통신 데이터 오버런을 나타낸다.

• 비트 0

상대로부터 데이터를 수신하였을 때 8250 LSI는 그 데이터를 수신버퍼 레지스터에 저장한다. 상태레지스터의 이 비트는 수신된 데이터가 수신버퍼 레지스터에 옮겨질 때 세트되며 데이터가 도착하여 읽혀갈 준비가 되어 있다는 상태를 나타낸다.

RS-232C 케이블의 접속

데이터통신에서 정보처리를 위한 기구를 우리는 흔히 DTE(Data Terminal Equipment)라고 부르며 이러한 정보처리기구를 실제의 송수신 케이블과 연결해 주는 장치를 우리는 DCE(Data Communication Equipment 혹은 Data Circuit Termination Equipment)라고 부른다. 즉, 컴퓨터가 DTE에 해당하며 컴퓨터와 송수신 회선을 연결해 주는 장치인 모뎀이 DCE에 해당한다고 할 수 있다.

그러나 컴퓨터끼리, 혹은 컴퓨터와 단말기를 직접 접속하는 경우 상황은 약간 달라진다. 즉, 컴퓨터나 단말기를 사용하는 사람의 입장에서 보면 자기의 컴퓨터나 단말기가 DTE이고 상대방의 컴퓨터를 DCE로 보면 그다지 문제가 없을 것이다.

RS-232라고 하는 것은 컴퓨터나 단말기내의 병렬 데이터를 외부로 통신하기 위하여 2진 직렬 데이터로 변환하여 송수신하는, 이와 같은 DTE 및 DCE의 데이터, 제어신호 및 타이밍 등을 규정하는 일종의 표준규격이다. 이러한 접속 규격을 규정함으로써 독자적인 방법의 구현에서 오는 혼란을 피하고 상호간의 원만한 통신을 가능하게 하는 직렬통신의 규격이다.

RS-232는 위에 기술한 DTE와 DCE간에 데이터를 주고 받기 위해, 코넥터의 형태나 핀번호, 각 핀의 신호에 대한 명칭 및 각 신호의 전기적 레벨 등을 규정하고 있으며 그 규정이 처음 제정된 이래로 두번의 교정이 가해져서 RS-232A, RS-232B, RS-232C로 변천해 왔으며 오늘날에는 RS-232C가 대부분 사용되고 있다. 표 1은 RS-232C 표준인 EIA 25핀 코넥터의 핀번호 및 가 핀에 해당하는 신호명을 나타내고 있으며 그림 5는 이 RS-232C 표준규격에 따른 DTE와 DCE의 접속예를 보여 주고 있다.

모뎀제어/상태 신호를 조금만 더 알자

표 1을 보면 알 수 있듯이 RS-232C 표준신호 규격에는 데이터 전송을 위한 Tx, Rx의 신호 외에도 다수의 신호가 있음을 알 수 있다. 데이터 송수신을 위해 필요한 신호는 Tx, Rx의 두가지의 신호만으로도 충분하지만 두대의 컴퓨터가 멀리 떨어져 있는 경우에는 모뎀을 사용하여 연결한다는 것을 독자 여러분은 알고 있을 것이다.

〔표 1〕 RS-232C 각 핀번호의 신호명

핀	신 호 명	입출력	핀	신 호 명	입출력
1	Protective ground	—	14	NC	—
2	Transmitted data	출력	15	NC	—
3	Received data	입력	16	NC	—
4	Request to send	출력	17	NC	—
5	Clear to send	입력	18	NC	—
6	Data set ready	입력	19	NC	—
7	Signal ground	—	20	Data terminal ready	출력
8	Received line signal detected	입력	21	NC	—
9	NC	—	22	Ring indicater	입력
10	NC	—	23	NC	—
11	NC	—	24	NC	—
12	NC	—	25	NC	—
13	NC	—			

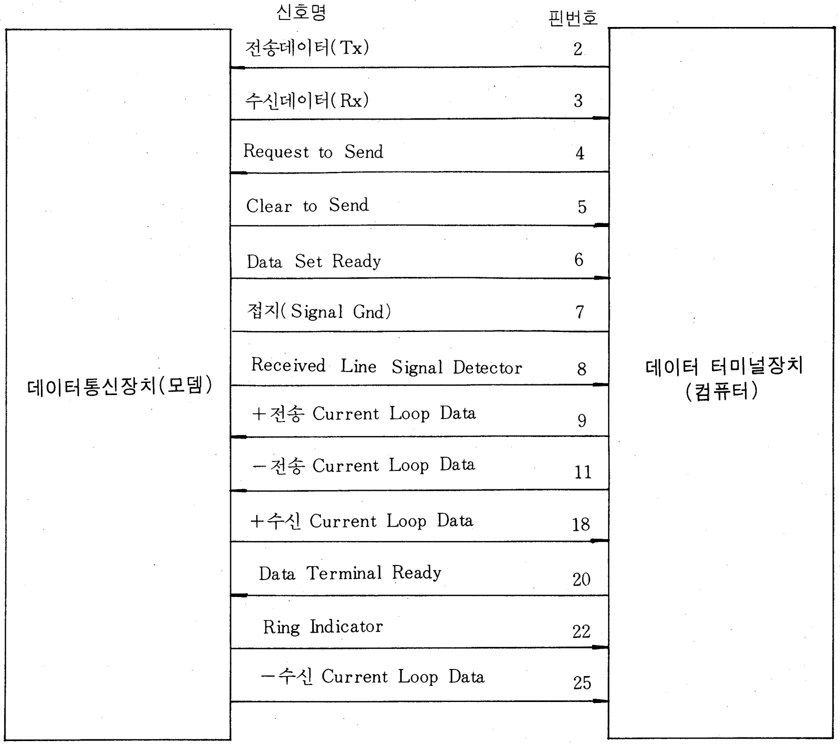
RS-232C의 표준규격에 의하면 두 컴퓨터간의 최대 송수신거리를 50피트로 제한하고 있으므로 (실제로 이 이상의 거리에서도 사용이 가능하다) 이 이상의 거리로 떨어져 있는 두대의 컴퓨터를 연결하는 방법이 바로 모뎀을 이용하는 것이다.

모뎀은 영어의 변조(MOdulation)와 복조(DEModulation)의 머리 글자를 따온 단어로 송신측의 입장에서 보면 컴퓨터에서 발생하는 디지탈신호를 전화선과 같은 통신매체를 통하여 신호를 전달할 수 있도록 아날로그신호(음성과 같은 신호)로 변환하는 역할을 한다. 데이터를 수신하는 입장에서 보면 이와 반대로 아날로그신호로 전달된 신호를 디지탈신호로 변환하는 역할을 한다(그림 6 참조).

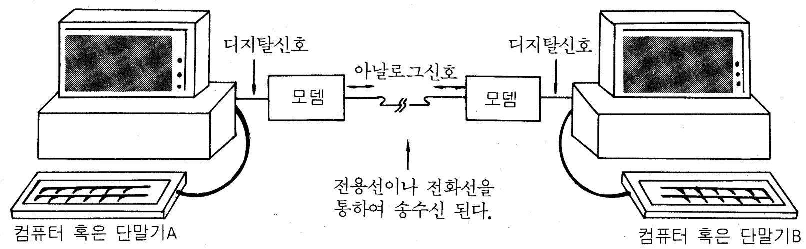
실제로 두대의 컴퓨터가 통신을 하는데 필요한 모든 데이터는 Tx, Rx 두가닥의 선을 통하여 모두 전달되지만 그 외의 다른 신호들은 컴퓨터와 모뎀, 즉 DTE와 DCE의 연결상태 혹은 DCE와 DCE끼리의 연결상태를 나타내는 신호로 사용된다. 이들 신호의 의미는 다음과 같다.

- **CTS(Clear to Send, 입력)** : 모뎀 제어 입력신호로 모뎀이 출력대기 상태에 있음을 나타낸다.
- **DSR(Data Set Ready, 입력)** : 통신링크(Communication Link)가 완료되어 데이터전송 준비상태에 있음을 나타낸다.
- **DCD(Data Carrier Detect, 입력)** : DTE와 연결된 모뎀이 상대방의 모뎀으로부터 데이터 캐리어(Data Carrier)를 검출하여 두 모뎀이 연결되었음을 나타낸다.
- **RI(Ring Indicator, 입력)** : 모뎀이 전화의 벨신호를 검출하였음을 나타낸다.
- **DTR(Data Terminal Ready, 출력)** : 모뎀에게 DTE가 통신 준비상태에 있음을 알려주는 신호이다.
- **RTS(Request to Send, 출력)** : 모뎀에게 DTE가 데이터를 보낼 준비가 된 것을 알려주는 신호이다.

〔그림 5〕 DTE와 DCE의 접속 및 신호방향



〔그림 6〕 모뎀과 모뎀의 연결



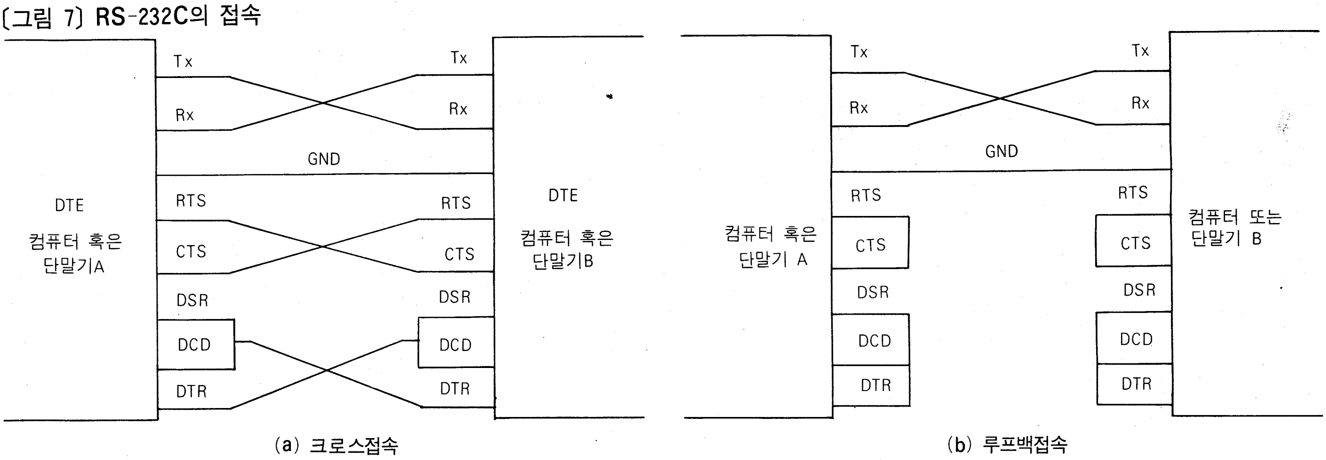
기타, 모뎀의 상세한 기능이나 규격에 대하여는 다른 서적을 참고하기 바란다.

컴퓨터를 연결한다.

이제까지 RS-232C의 규격 및 그 신호들에 대한 정의를 알아보았다. 이제부터는 실제로 RS-232C 직렬포트를 통하여 다른 컴퓨터나 단말기, 프린터 등을 연결하여 보자. 컴퓨터를 직렬포트를 이용하여 이들을 처음 연결하여 보려는 초보자의 경우 메이커에서 제공하는 케이블로 이들을 쉽게 연결할 수 있지만, 코넥터의 형상이나 케이블의 가닥 수가 달라지면 이러한 접속에 당황하는 경우를 왕왕 보게 된다. 그러나 RS-232C의 각 핀에 해당하는 신호의 정의와 그 신호들의 방향에 대한 지식을 바로 가지고 있으면 어떠한 상황에서도 직렬포트를 쉽게 연결할 수가 있다.

한가지 예로 모뎀을 사용하지 않고 전송 케이블을 이용하여 두개의 컴퓨터를 연결해야 할 경우를 들어본다. 이

〔그림 7〕 RS-232C의 접속



를 위하여 가장 먼저 해결해야 할 문제는 데이터의 전송방향을 이해하는 것이다. 25핀 코넥터(PC-XT)나 9핀 코넥터(PC-AT는 대부분 9핀 코넥터를 제공한다)중 어떠한 형태의 코넥터를 사용하더라도 신호의 방향만 알면 아무런 문제가 없다.

먼저 데이터 선들, 데이터 출력(Tx)/데이터 입력(Rx) 선을 연결한다. 자신이 사용하는 컴퓨터의 데이터 출력 Tx신호는 상대방에서 바라보면 데이터 입력신호 Rx가 되므로 이들을 상호 교차하여 연결한 다음, 신호 접지선(Signal Ground)을 연결한다. 이들 세 선의 연결만으로도 기본적인 연결은 완료되었으며 이러한 연결상태에서 대부분의 통신용 소프트웨어를 사용할 수 있다.

그러나 직렬포트를 이용한 DOS의 COPY 명령이나 일부 통신용 소프트웨어는 이 세가닥 선만의 연결로는 충분하지 못하다. 예를 들어 아래에 열거한 명령을 DOS 프롬프트하에서 입력하였다고 하자.

```
A> MODE COM1:96,N,8,1
A> COPY TESTFILE COM1
```

이 예는 직렬포트 1(COM1)을 9600 BPS, 8비트 데이터 길이, 1개의 스톱비트를 패리티비트를 사용하지 않고 초기화하는 명령이다. 다음으로 TESTFILE 이라는 샘플 텍스트를 직렬포트 1을 통하여 전송하고자 함을 나타낸다. 그러면 사용자는 다음과 같은 메시지를 보게 될 것이다.

```
Write fault error writing device COM1
Abort, Retry, Ignore, Fail ?
```

분명 케이블의 접속에는 아무런 이상이 없었다. 그런데 왜 이런 현상이 나타나는가? 그것은 8250 ACE의 내용 규격을 보면 이해되겠지만 간단히 말해 필요한 신호가 연결이 되지 않아서 컴퓨터내의 8250 칩이 상대방의 컴퓨터가 준비상태에 있지 않은 것으로 판단하기 때문이다.

이러한 문제는 위에 열거한 신호를 상대방의 신호와 적당히 연결하여 주면 된다. 원래 이들 신호는 모뎀상태나 제어를 위한 신호지만 지금과 같이 컴퓨터를 직접 접속하는 경우에는 모뎀의 제공하는 출력신호인 이들 신호를 사용할 수 없으므로 상대방 컴퓨터의 출력신호를 이용하여 당신이 가지고 있는 컴퓨터를 슬쩍 속이면 된다.

즉, 상대방의 출력신호를 당신이 가지고 있는 컴퓨터의 입력신호로 이용하면 된다. 구체적으로 설명하면 모뎀 제어/상태에 상응되는 신호들을 연결하여 주면 된다. 표 1에서 보면 알 수 있듯이 당신이 사용하고 있는 컴퓨터의 출력신호는 RTS, DTR이고 필요로 하는 입력신호는 CTS, DSR, DCD 등이다. 따라서 상대방의 RTS 출력신호를 당신의 CTS 신호에, 상대의 DTR 출력신호를 당신의 DSR, DCD에 연결하면 된다. 이를 그림으로 표시하면 그림 7(a)와 같다.

이와 같은 연결방법은 상대의 컴퓨터나 단말기가 위에 열거한 모든 신호를 제공하고 또 이 모든 신호를 사용할 수 있는 케이블로 연결되어야만 실현 가능하다. 실제로 이들 신호를 모두 사용하기 위해 필요한 선의 가닥수는 최소한 7가닥 이상이어야 가능하다. 이러한 연결방법은 이들 모뎀 제어/상태 신호를 이용하여 데이터의 흐름을 직접 조절하는 하드웨어적인 데이터제어 방법에 이용되기도 한다.

그런데 독자가 가지고 있는 케이블이 4가닥이나 그 이하의 케이블 밖에 되지 않는다면 어떻게 해야 할까. 이러한 문제도 위의 방법을 약간 응용하면 해결할 수 있다. 즉, 독자가 가지고 있는 컴퓨터의 출력신호를 입력신호에 연결하는 방법이다. 즉, RTS 신호를 CTS에, DTR 신호를 DSR 및 DCD 신호에 연결하는 방법이다. 이상하게 보일지 모르지만 아무런 문제가 없으며 실제로 많이 사용되는 방법이니 안심하기 바란다. 이러한 방법을 흔히 루프백(Loop-Back)이라고 하며 최소한의 케이블 연결로 통신을 가능하게 한다.

PC입문 1

초보자의 PC 입문

이정섭 / 숭실대학교 정보과학대학원

파스칼*의 효심으로부터 출발한 컴퓨터는 이제 인간을 대신해서 계산을 수행하는 단순한 기능뿐만 아니라 첨단의 정보화 사회에서는 빼놓을 수 없는 문명의 이기로서 자리를 굳건히 하고 있다. 그 힘이 얼마나 강한지 요즘 들어 많은 분들이 학교에서나 학원에서 혹은 아는 사람을 통하여 컴퓨터에 관심을 갖고 배우려고 노력하는 모습들을 여러 곳에서 볼 수 있다.

필자는 가끔 '컴퓨터가 그렇게 매력적인 것일까?'라고 생각해 볼 때가 있다. 그때마다 글쎄(?)란 답이 떠오르곤 하였는데 여러분께서는 어떠실른지? 아마도 처음 컴퓨터를 접하시는 분들은 컴퓨터의 매력이란 단어 자체의 의미에 대해 이해하기 어려울 것이다.

컴퓨터에 대한 철학적인 답은 차차 생각해 보기로 하고 본지를 통해 여러 분들과 함께 현재 사용중인 PC(8비트, 16비트)에 대해 고찰해 볼 기회를 갖게 된 것을 무척 기쁘게 생각한다. 앞으로 다루어질 내용은 컬럼 성격을 띠면서 초보자와 경험있는 사용자들이 보아도 부족하지 않을 만한 부분도 다루고자 한다.

연재를 시작하면서

시내의 대형 서점가를 나가게 되면 마치 참새가 방앗간을 그냥 지나가지 못하듯 꼭 들르는 곳이 컴퓨터 서적코너이다. 이곳에 이르면 언제부터인지 컴퓨터를 처음 배우려는 사람들을 위한 마땅한 책이 있는가 살펴보는 그런 습관 때문에 컴퓨터에 대한 여러 신간 서적을 유심히 살펴보게 되었다. 그것은 컴퓨터를 처음으로 배우고자 하는 사람과 초보자들이 부딪치는 모든 문제들을 해결할 수 있는 방법이 없을까 하는 생각에서 였다.

그러나 이 모든 문제를 한번에 해결하기에는 내용이 빈약하거나 아니면 어느 정도 초보자의 경지를 벗어난 분들을 위한 듯한 내용으로 되어 있어 수많은 내용의 책이 널려 있어도 초보자에게는 먼저 안 사람에 의해 자세히 설명 되어지지 않는 이상 단지 막연하게 컴퓨터에 대한 관련서적이 될 뿐 도움을 주지 못한다.

그리하여 그 차이나는 빈 공간의 간격에 다리 역할을 하는 지식이 필요하다고 생각하게 되었고, 컴퓨터를 배우고자 하는 분 그리고 현재 잘 사용하고 계신 분들께 조그마한 도움이 되고자 감히 이렇게 글을 쓰게 되었다.

우선 기존의 여러 서적에 비슷하게 나와 있는 내용보다는 그 서적에서 '이 글을 읽는 독자는 이 정도의 지식은 다 알고 있겠지'하고 설명을 생략한 부분이라든지 알기 쉽게 설명할 수 있는데 어렵게 설명해 놓은 것이라든지 등등 초보자가 기존의 책에 접근해서 널리 있는 지식들을 이용할 수 있도록 그 기초를 제공하는데 촛점을 두고자 한다.

앞으로 다루게 될 주제들을 간단히 살펴보면 다음과 같다. 이번 호에서는 본지를 통해 이야기의 촛점이 될 PC가 컴퓨터라는 범주 안에서 차지하는 위치에 대해서 알아보고 PC 마련의 기본사항 및 대부분 알고 있으면서도 실수하기 쉬운 컴퓨터를 아끼는 방법에 대한 상식적인 이야기를 할 것이며, 두번째는 주로 사용되는 PC의 하드웨어와 소프트웨어 그리고 매뉴얼을 적극적으로 이용하는 방법에 대해서 언급할 것이다.

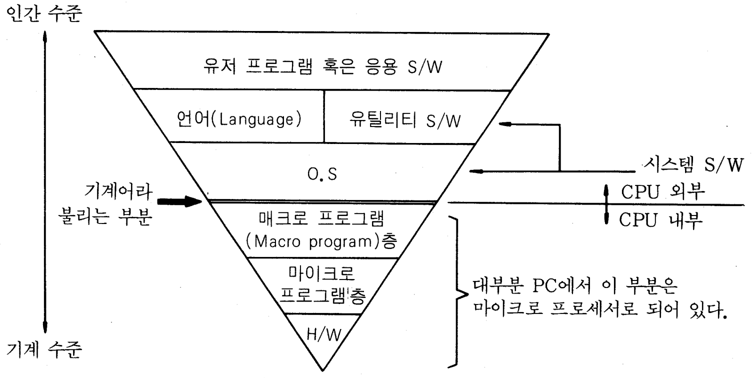
세번째는 **그림** 1에 나타낸 바와같이 맨 위의 층인 사용자 프로그램과 응용 프로그램이 개인적 차원에서 어떤 의미를 지니고 있는지 살펴보고 이의 응용에 대해서 생각해 볼 것이다.

네번째는 유틸리티에 대해서 그리고 그 다음부터는 PC의 OS와 PC 차원에서 많이 애용되고 있는 언어를 살펴보고 이의 발전방향에 대해 점검해 보도록 하겠다.

또한 프로그램 작성에 있어서의 노하우적인 면과 어떻게 하면 프로그램을 잘 작성할 수 있을까 하는 것도 토의가 될 것이다. 그리고 어셈블리 프로그램 및 기계어에 대한 개념적인 고찰을 CPU의 구조적인 측면에서 바라 보고자 한다.

*파스칼(Pascal Blaise 1623~62)
프랑스의 수학자, 물리학자, 철학자, 사형기하학에서 '파스칼의 정리'를, 유체정력학에서 '파스칼의 원리'를 발견했음.
그는 또한 지방 세금징수원으로 방대한 계산에 쫓기고 있던 아버지를 돕기 위해 수동식의 톱니바퀴 계산기를 만들었다.

〔그림 1〕 H/W, S/W의 전체적 성격상 분류



처음 배우려는 분들을 위하여

속담에 '시작이 반이다'라는 말이 있듯이 용기있는 결단으로 빨리 컴퓨터에 대한 배움을 시작하라고 말씀드리면서 배우긴 배워야 할텐데 하고 마냥 생각만 하게 만드는 원인을 파헤쳐 더이상 컴퓨터 문맹의 경지(?)에 계속 머무르지 않기 바란다.

컴퓨터를 처음 접하는 사람들은 미지의 것에 대한 두려움 혹은 거부감이 공통적인 현상으로 나타나는데 이러한 현상에 대한 여러가지 반응중에 가장 빈도가 높은 것 몇가지를 조사하여 보았다. 이 글을 읽고 있는 독자 자신은 본인이 어디에 해당되는지 비교해 보기 바란다.

우선 '컴퓨터에 대해선 완전백지인데(혹은 전혀모르는데), 그래도 컴퓨터를 배울 수 있겠습니까?', '컴퓨터에 대해서 제대로 배우려면 시간이 얼마나 걸립니까?', '컴퓨터는 도대체 어떤 것을 택해야(혹은 구입해야)됩니까?', '얼마에 사야지 속지않고 잘사는 것인지요', '컴퓨터를 배우려고 학원을 다닐까 하는데 어떨지요', '컴퓨터를 배우고 있는데 무슨 말을 하고 있는지 도무지 모르겠습니다. 어찌 하면 좋은지요', '남들이 많이 배우길래 나도 빠지면 안될 것 같아서 배우고 있는데 너무 어려워서 뭐가뭔지 모르겠습니다. 좀더 쉽게 아는 방법이 있는지요?' 등등의 수백수천가지의 별의별 희한한 의문사항들이 꼬리를 물고 쏟아지면서 힘들게 한다.

그 다음으로 의욕상실증이 있다. 이것은 배우고 싶은 욕망이 주체할 수 없이 솟아 오르다가 조그만 어려움이 닥쳐오면 제풀에 꺽여 버리는 현상이 나타나는데 이 현상의 예방책은 지속할 수 있는 인내력이라 할 수 있다. 그리고 대부분의 문제들을 보면 시간, 돈, 노력에 대한 상관관계에서 발생된다. 그러므로 이러한 문제에 대해 복잡하게 생각하기 보다 실제로 차분하게 시작할 것을 당부하고 싶다.

컴퓨터 범주 안에서의 PC의 위치에 대하여

대부분의 초보자들이 컴퓨터하면 어렵다고 생각하는 이유가 선입견 때문이다. 이러한 문제점의 발생은 다름아닌 매스미디어의 단점중의 하나라고 볼 수 있다. 쉽게 설명해서 컴퓨터하면 먼저 우주선을 발사하는 장면이나 호호백발의 과학자가 매우 복잡한 기계 앞에서 연구에 몰두하거나 첩보영화 같은데 나오는 컴퓨터실의 장면을 그대로 떠올릴 수 있는데 이것은 생각할 기회를 주지 않고 그대로 무의식중에 단순화된 의식을 심어주기 때문이다.

이때문에 컴퓨터를 대할때 어렵게 생각되며 거부반응부터 일어나는 것이다. 그럼 이 어렵다고 생각되는 거부반응 증세부터 치료하기 위해 옛날 컴퓨터가 대중화되기 이전시대와 대중화된 이후시대를 비교해 보고 생각을 정리해 보자.

옛날 대중화되기 이전 시대의 컴퓨터는 오늘날 대부분의 초보자들이 막연하게 알고 있는 바로 그것으로 주로 연구소나 특수한 기관에서 많은 지식과 경험을 쌓은 과학자들이 사용하는 것이었고 이의 사용에 있어서도 대단히 어렵고 복잡했기 때문에 일반사람들은 구경할 수도 없었으며 더더욱 만져 볼 수도 없었다. 그저 계산하는 것이나 일을 처리하는데 인간보다 정확하고 빠르게 처리해 낸다는 사실만 상식적으로 알려져 있을 따름이었다.

이것이 컴퓨터를 처음 배우려는 분들이 배우기 어렵다고 생각하도록 만드는 근본원인이라 생각된다. 그러므로 컴퓨터가 대중화된 이후의 바뀐 상황에 대해서 이해한다면 자연히 그 증세는 없어지리라 생각된다.

개인이 컴퓨터-정확히 표현하면 PC를 소유할 수 있게 된 주원인은 전자공학중에서도 반도체 기술의 발전이었다. 그로 인해 컴퓨터가 소형화, 경량화, 저가격화 되었는데 이때문에 옛날 고전적인 생각에서 수정해야 할 많은 것들이 생기게 되었다.

그 중의 하나가 초보자들의 의문점 중의 하나인 성능대 소형화와 가격비이다. 옛날에는 크기가 커야지 성능이 좋았으나 지금은 32비트 컴퓨터가 무릎위에 올라갈 만큼 소형화 되었으며 성능은 거의 비슷하거나 옛날 것에 비해 나아졌다. 따라서 옛날의 생각은 수정되어야 한다.

그러면 개인이 사용하는 컴퓨터와 연구소등에서 과학자들이 사용하는 컴퓨터는 어떤 관계에 있는가? 그릇에도 용도에 따라 다양한 크기와 모양과 색깔이 있듯이 컴퓨터에도 사용되는 용도나 그것을 사용하는 사람에 따라 천

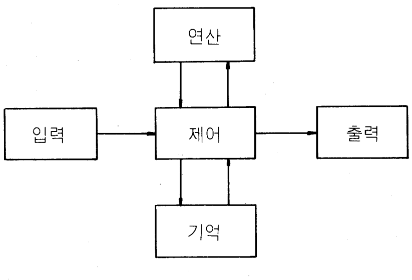
차만별의 종류가 있음을 기억하기 바란다.

그러므로 개인이 사용하는 컴퓨터나 과학자가 사용하는 컴퓨터는 같을 수도 있고 전혀 다를 수도 있는데 다른 것이 대부분이다. 왜냐하면 연구에 사용되는 컴퓨터는 보다 빠르고 정확해야 하기 때문이다. 그러면 개인이 사용하는 컴퓨터는 느리고 나쁜 것인가? 절대로 그렇지 않고 옛날 초창기의 미니 컴퓨터의 성능과 비슷하다고 볼 수 있기 때문에 개인이 이만한 성능의 컴퓨터를 사용할 수 있게 된 것도 축복이라 생각해야 한다.

이야기가 약간 다른 방향으로 흐른 것 같은데 컴퓨터에는 개인이 사용하든 연구소에서 과학자가 연구용으로 사용하든지 간에 기본적인 구성요소가 있는데 이중에 하나라도 빠지면 컴퓨터라 칭하기에는 좀 고려해 보아야 할 것이다. 그림 2에서 볼 수 있는 것처럼 컴퓨터의 기본 구성요소는 입력장치, 연산장치, 기억장치, 제어장치, 출력장치인데, 책상위에 올라 가는 데스크 탑이나 무릎위에 올라가는 랩탑 컴퓨터도 이 기본 구성요소를 전부 갖추고 있기 때문에 당연히 컴퓨터라 칭하고 있고, 성능도 32비트나 16비트이기 때문에 우습게 볼 수는 없다. 따라서 앞으로는 컴퓨터가 요즘의 전자 손목시계처럼 손에 차고 다닐 수 있는 시기가 와도 전혀 이상하지 않을 것이다.

컴퓨터의 종류를 살펴보면 계산능력이 가장 빠른 수퍼컴퓨터를 비롯하여 대형, 중형, 소형 그리고 미니컴퓨터, 마이크로컴퓨터, 퍼스널컴퓨터 등이 있는데 우리가 쉽게 접근할 수 있는 컴퓨터는 개인이 사용할 수 있는 정도의 가격과 성능을 가지고 있는 퍼스널 컴퓨터(PC)이다. 따라서 다루기 힘든 커다란 컴퓨터를 배우는 것이 아니므로 미리 '어려울 것이다'라는 선입견을 버리기 바란다.

〔그림 2〕 컴퓨터의 5 대장치



PC마련의 기본사항

PC의 자세한 구성에 관하여는 다음에 하드웨어와 소프트웨어의 설명에서 좀더 자세히 살펴보기로 하고 이번호에서는 PC를 마련하기 위한 기본적인 사항을 알아보자. PC는 8비트와 16비트로 나누어서 이야기할 수 있는데 대부분의 8비트는 APPLE기종과 MSX 그리고 Z80 CPU를 사용한 기종으로 볼 수 있다. 이것을 어느 정도 쓸만하게 사용하려면 적어도 본체와 모니터 그리고 플로피디스크 드라이브 2대 정도가 필요하게 된다. 그리고 프린터도 있으면 좋겠다.

요즘 대부분의 판매점에 나가보면 8비트보다는 16비트 PC/XT 정도가 대부분의 구입자들에 의해 선호되고 있다. 16비트중에 IBM PC 호환기종들을 보면 메이커에서 생산된 것들과 청계천업체에서 만들어진 것 그리고 대만제 수입품이 있는데 어느 것이든지 실제로 알맞게 쓸만한 정도가 되려면 640KB 이상의 메모리와 본체 그리고 키보드, 모니터(특정한 용도로 쓰이는 컬러모니터 이외에는 대부분 단색으로 모노크롬이라 불리는 모니터를 사용한다.)와 플로피 디스크드라이브 2대, 비디오 카드로는 모노크롬일 경우 허큘리스 카드에 한글카드가 장착된 것이 필요하다.

이것이 PC를 마련하는 가장 기본적인 사항이며 여기에 하드디스크 드라이브와 마우스 그리고 프린터가 있으면 편리하게 사용할 수 있다. 이상의 조건을 잘알고 구입하려는 경우 신중하게 해야 할 것이다. 왜냐하면 엎질러진 물은 다시 컵에 담을 수 없듯이 구입해 놓고 후회해 보아야 소용이 없는 것이기 때문이다.

컴퓨터를 아끼는 상식

컴퓨터의 입문자가 알아야 할 상식 몇가지를 알아보자.

첫째, 컴퓨터의 명칭 문제인데 대부분의 사람들이 컴퓨터와 포켓계산기를 혼동해서 부르고 있다. 영어로 이야기하면 구분이 확실하게 되는데 컴퓨터는 Computer이고 포켓계산기는 Calculator이다. 그러므로 둘다 똑같이 컴퓨터나 계산기로 부르는 착오가 없기 바란다.

둘째, 자기 몸처럼 컴퓨터를 아끼라는 것이다. 이것은 컴퓨터의 수명문제와 직결되기 때문에 반드시 아끼는 습관이 있어야 한다. 내것이 아니라 다른 사람의 것이라도 내 물건처럼 아껴서 다루길 바란다. 그러면 컴퓨터도 자기를 아끼는 사람에게는 함부로 심술(?)을 부리지 못할 것이다.

일반적인 경우 잊어버리기 쉬운 것이 하드디스크를 사용했을 때 헤드파킹에 관한 것이다.

대부분의 초보자들은 이것을 잊고 전원스위치를 끄기 때문에 다음 사용시에 문제가 발생하는데 왜 그런지 그 이유에 대해서는 자세히 설명되어 있는 기존의 글을 참조하시기 바라고 반드시 하드디스크 사용자는 전원을 끌 때 헤드파킹을 시키는 습관을 몸에 익히기를 바란다.

세째, 무리하게 컴퓨터를 장시간 운용하자 말라는 것이다. PC

교육용 PC, 우여곡절 끝에 16비트로 확정

그동안 기종선택 문제로 많은 논란의 대상이 되었던 교육용컴퓨터가 우여곡절 끝에 16비트로 확정되었다.

정부가 교육용 기종을 16비트로 전환하게 된 첫째 이유는 가격문제 때문이라고 보인다. 16비트 컴퓨터는 근간에 계속 가격이 하락되고 있는 추세인데 비하여 8비트 컴퓨터는 가격이 하락될 전망이 불투명하다는 점이다.

둘째는 호환성문제이다. 국가기간전산망에 설치되는 컴퓨터가 행정전산망표준규격에 따라 16비트 퍼스널컴퓨터와 호환성을 유지하기 때문에 국내전산망의 호환성과 정보화의 촉진을 위하여 16비트 기종을 선택하는 쪽이 낫다는 인식이 많은 실무자들 사이에서 대두되었다.

세째는 프로그램 개발비용과 교육의 일관성이다. 8비트 컴퓨터를 선택시 컴퓨터 기종간의 호환성 문제로 인한 프로그램 개발비용의 증가와 학생이나 교사의 이동에 따라 발생하는 교육의 일관성에 관한 문제등의 전례로 교육용 컴퓨터의 표준화 작업이 필요하게 되었기 때문이다.

또한 표면으로 드러나지는 않았지만 변수로 작용되었다고 생각되어지는 것은 상공부, 과기처, 체신부 및 컴퓨터 연구조합에 가입된 컴퓨터업체들이 호환성 및 국내산업육성을 위하여 표준화된 교육용 컴퓨터를 채택할 것을 강력하게 주장하였다는 점이다.

16비트 컴퓨터의 보급을 위한 정부의 대책으로는 관련기관들의 표준화방안과 컴퓨터기기의 저가공급방안, 문교부와 체신부의 공동 예산지원, 그리고 교육용 소프트웨어 개발보급방안 등을 골자로 하고 있다.

표준화 방안의 절차를 살펴보면 문교부와 한국교육개발원이 주축이 되어 이들 산하의 교육연구망추진위원회와 전산망조정위원회에서 규격을 의결, 이를 문교부에서 공지하고, 공급업체에서 제작된 컴퓨터를 한국교육개발원 주관하에 전문기관에서 기능 및 호환성을 시험평가하여 최종적으로 문교부에서 합격기종을 통지하여 기종을 선정케 되어 있다.

그리고 상공부와 컴퓨터연구조합은 표준화를 통한 퍼스널컴퓨터의 대량생산 및 저가공급을 유도하고 있다.

또 예산지원방안은 체신부 산하의 전기통신공사가 공중전화의 낙전금을 이용하여 국민학교에 연차적으로 컴퓨터를 무상대여 형식으로 지원하면서 정비보수의 책임까지 지기로 하였고, 중·고등학교에는 문교부에서 예산지원을 하기로 하였다.

그리고 컴퓨터교육 확산에 따른 전문인력의 수요에 대응하기 위한 방침으로는 현재의 교육대학이나 국·사립사대, 관련대학에 컴퓨터교과과정을 설치하여 각각 필수적으로 학점을 이수하도록 할 방침이며, 전자계산학을 부전공으로 양성, 적극 권장할 계획이다.

이와 함께 기존의 교사들을 재교육하는 방안으로는 교양연수, 자격연수, 일반연수, 전문연수 등의 부문으로 나누어 대상교사를 선정하고, 연수내용을 설정하여 그에 따른 연수방법이 발표되었으며, 올해에 3천6백명의 컴퓨터교육 담당교사를 대상으로 해외연수과정을 거치게해 교사의 자질향상에 기여하는 방안도 마련되었다.

이같은 정부발표는 단지 컴퓨터의 교육적인 측면뿐 아니라 컴퓨터의 저변확대를 통하여 전국 네트워크망 시대의 구축을 위한 기초작업이기도 하다.

그리고 16비트 기종은 국제적으로도 업무·연구용은 물론 교육용으로도 널리 이용되고 있으므로 학교교육의 사회로의 연장이라는 측면에서도 매우 바람직한 일이라 할 수 있다. 정부의 이번 방침이 물론 여러가지 타당성을 고려한 신중한 검토끝에 결성된 것인만큼 지금이라도 시행변경을 하게 된 것이 다행한 일이긴 하지만 한편으로는 그동안 컴퓨터교육을 위해 투자된 8비트 퍼스널컴퓨터 기종에 관한 손실과 소프트웨어 개발에 관련된 인력과 시간에 따른 경제적 손실을 감안할 때 또다시 이러한 시행착오가 되풀이되는 일이 없어야 할 것이며, 이번 발표가 애초에 뜻한바 대로 이루어져 부디 국가백년지대계를 위한 참신한 실효 효과를 거둘 수 있기를 바랄 뿐이다.

덧붙여, 앞으로 16비트 퍼스널컴퓨터 공급업체들은 제품의 판매에만 신경을 쓸 것이 아니라, 현재 전무상태인 16비트 퍼스널컴퓨터의 교육용 소프트웨어의 개발부문에도 한 몫을 담당해야 할 것으로 생각한다.

〈글·이규화기자〉

국내소식

—상공부
90년부터 EWS 개발키로

상공부와 컴퓨터 관련업체에서는 90년부터 5년 동안 공동으로 EWS(엔지니어링워크스테이션)을 개발하기로 하였다.

이 EWS의 개발계획은 제1단계로 외국기종을 국내 환경에 맞는 개량 기종으로 개발하고, 제2단계로 1단계에서 축적된 기술을 바탕으로 국산 기종을 개발하며, 제3단계로 부가가치가 높은 독자적인 EWS 기종을 개발한다는 내용으로 구성되어 있다.

EWS는 고성능 워크스테이션으로 미세한 전자회로의 설계에서부터 공장자동화에 이르기까지 산업전반에 걸쳐 다양하게 사용되는 컴퓨터이다. 그러므로 EWS의 개발은 컴퓨터기술의 고도화와 품질향상에 있어서 필수적이다.

—체신부
첨단 통신단말기술 개발 착수

체신부는 오는 90년 이후의 고도화된 정보사회에 대비하여 올해부터 첨단 통신단말기술 개발을 착수한다.

이 개발에 따른 단계별 계획은 다음과 같다.

• 1단계(89~91년) : 텔리텍스 및 문자형 단말기술, G4 고속팩시밀리, 화상전화 등 비음성서비스 단말기술 개발
• 2단계(92~2001년) : 매체간 변환기술, 음성과 영상 압축기술, 음성과 문자정보 입출력기술 등 복합적이고 광대역적인 서비스가 가능한 단말기술 개발
• 3단계(2001~2011년) : 인공지능형 종합워크스테이션, 영상정보 입출력기술, 고기능 서비스단말 기술 개발

이들 단계를 거쳐 최종적으로는 ISDN(종합정보통신망)서비스를 활성화할 방침이다.

—한국전기통신공사
전화망 이용 공중 VAN 활성화

한국전기통신공사는 PSTN을 기초로 정보통신네트워크를 구축하는 한편 정보이용과 관련된 전화번호, 과금체계 등을 별도로 마련하여 정보이용확산의 기반을 조성함으로써 전화망을 이용한 공중 VAN을 크게 활성화할 방침이다.

이에 따라 한국전기통신공사는 각종 생활정보 데이터베이스를 체계적으로 '700'국에 분리 수용하여, 이용편의를 높일 계획인데 그 초기단계로 올 하반기부터 서울과 부산지역에 '700'번 서비스망을 구축하여 서비스를 실시할 계획이다.

또한 90년 이후부터는 PC, FAX, 비디오텍스 등을 통해서도 이용할 수 있도록 할 계획이다. 이렇게 되면 다양한 단말기로 데이터베이스 이용이 가능하게 되어 서비스 선택범위가 넓어지게 되고, 데이터베이스 번호체계의 통일로 서비스 이용이 편리해지며, 통화완료율이 높아져 정보 이용이 용이해진다.

뿐만 아니라 이 공중 VAN이 활성화될 경우, 통신공사측도 특정정보 이용량 폭주로 인해 공중망에 미치는 영향을 줄일 수 있으며, 새로운 서비스 수요창출로 수익을 증대시킬 수 있을 것으로 전망된다.

—한국전기통신공사
올 상반기중 데이터통신 시범망 구축완료

한국전기통신공사는 오는 91년 말까지 시범적으로 사용될 데이터통신 시범망을 빠르면 올 상반기 중에 구축을 완료하고 하반기부터 실제 운용에 들어갈 예정이다.

이번 시범망 구성은 전국 도청소재지급 이상의 대도시지역 전자식 시내교환기에 시외착신 전환기능을 부착하여 데이터전송 사업용 전화를 지정함으로써 구축된다.

이 시범망의 이용방법은 전자교환집중국통화권내에 있는 이용자가 사업용 전화를 사용하면 특정 데이터베이스에 시외접속되고 사용료는 시내 전화요금이 적용된다.

이 시범망 이용 대상은 우선적으로 통신공사에서 시행하고 있는 데이터통신 시범사업에 참여하면서 장거리 시외요금으로 접속하는 사업자와 전기통신공사와 특정 역무제공업자 협정을 맺은 업체나 개인에게 적용된다.

—한국전기통신공사
한국형 종합정보망(ISDN) 구축

한국전기통신공사에서는 발신자 전화번호 확인과 정지화상전화 등의 한국형

종합정보망을 국내기술로 전화선에 도입하였다. ISDN은 전화회선을 이용, 전기신호를 디지탈신호로 변환하여 다중채널방식으로 정보를 교환할 수 있는 체계이다. 전전자교환기에 ISDN 기능을 부가하여 텔리텍스나 고속팩시밀리 등의 첨단단말기도 연결하여 사용할 수 있다. 그리고 ISDN은 디지탈 신호를 세 가지로 구분하여 각각의 신호에 정보를 실어 보낼 수 있어 하나의 전화선으로 세 가지 정보 교환이 이루어지는 셈이다. 신호는 고채널(B) 2개, 저채널(D) 1개로 고채널은 음성이나 화상 등의 정보를 고속으로 전송할 수 있고, 저채널은 각종 신호와 데이터를 저속으로 전송하는데 사용된다.

ISDN을 이용한 회선은 3대의 단말기를 동시에 사용할 수 있으며, 최대 8대의 단말기를 접속하여 대량의 정보를 전송할 수 있는 잇점이 있다. 이중 D 채널은 전화를 받을때 수신자의 전화기 표시판에 발신자의 전화번호를 표시할 수 있으며, 국제전화 통화시 사용시간과 요금의 표시도 가능하다. 홈오토메이션에서는 원격조정에 의한 감시나 계기의 자동점검, 수신자 요금부담, 신용카드 호출서비스 등 다양한 기능으로 활용할 수 있다. 발신자 전화번호가 표시되는 전화가 사용되면 익명의 장난전화 등을 방지할 수 있지만, 한편으로는 제보나 고발 등을 하는 사람들에게 신분보장을 할 수 없다는 우려가 있다.

다음은 ISDN의 주요 지원내용이다.

• 전화기 : 발신자의 전화번호나 요금확인 등을 액정판에 표시하며, 기존 전화기도 변환기를 부착하여 사용이 가능하다.

• 정지화상전화 : B채널 두개중 하나는 음성정보를 다른 하나는 화상정보를 송수신하여 상대방의 모습을 정지화면으로 보면서 통화할 수 있다.

• 텔리텍스 : 워드프로세서의 기능을 텔렉스에 추가한 것으로써 1만자 정도를 2~3초내에 전송할 수 있다.

• 퍼스컴통신 : 토목공사나 자동차, 전자회로 등 복잡한 설계도와 자료 등의 대량정보를 고속으로 전송이 가능하다.

• 고속팩시밀리 : ISDN의 B채널을 사용하여 1장당 2~3초로 전송시간이 단축되어 비용을 줄일수 있다.

—한국전자통신연구소

기본형 단말기 표준규격안 마련

전자통신연구소는 단말기의 기본형을 16비트 XT기종으로 하는 표준규격안을 작성하여 체신부와 한국전기통신공사에 전달한 것으로 지난 6월 3일 알려졌다.

정부는 또 오는 2007년까지 정보 이용 활성화를 위해 PC 1천만대 보급을 하는데 있어서 16비트 XT기종을 기본형 단말기로 선정할 전망이라고 한다. 지난해 체신부 업무보고에서 정보사회의 조기 정착이 거론된 이후 PC1천만대 보급 계획이 있었으나 그동안 진전이 없다가 이번 표준 규격안으로 국내 정보통신업계에 비상한 관심거리로 주목받고 있어 국내시장이 좀더 활발해질 전망이다.

전자통신연구소가 준비한 기본형 단말기 표준규격안은 2바이트 완성형의 KS한글코드와 VT220 터미널모드, 그래픽이 가능한 컬러모니터 기능 등을 갖고 있고, 이를 사용하여 데이터베이스와 각종 생활 정보 검색, 전자사서함, 비디오텍스기능 등을 이용할 수 있도록 했다.

체신부와 통신공사는 이 규격안을 토대로 올 하반기부터 공청회와 시제품 제작을 하고, 내년부터 보급할 계획인데 앞으로 지능형 단말기도 개발할 계획이다.

—동양나일론(주)

19개 VAR 사 사장단 간담회 개최

동양나일론(주)의 효성컴퓨터사업부(대표 : 공정곤)는 지난 6월2일 프리마호텔에서 제1차 효성컴퓨터 VAR사장단 간담회를 개최했다.

이날 간담회에서는 (주) 경시스템외 19개사 VAR가 모여 신규 VAR의 상견례 및 인증패(서) 전달식에 이어 활성화 방안과 향후 영업전략에 대한 내용으로 다음과 같이 토의되었다.

- POWER 88/286의 보급 확대방안
- POWER 386의 판매 증대
- POWER 88A(가칭)의 교육용 시장 공급전략

본사와 VAR간의 우호적인 유대관계와 협조체제 강화를 위해 마련된 이 모임은 앞으로도 계속 정기적인 모임을 가질 예정이다.

—삼성그룹

정보통신부문 구획정리 구체화

삼성그룹은 정보통신부문을 컴퓨터와 통신사업으로 양분할 방침으로 알려졌다.

삼성의 이러한 방침은 컴퓨터사업의 강화를 위한 것으로 통신기기 사업과 함께 정보통신 부분으로 운영되고 있는 조직을 컴퓨터와 통신으로 양분하여 계열기업의 컴퓨터 관련 사업을 흡수할 계획이다.

이같은 구획정리 방침에 따라 삼성전자에 컴퓨터 사업부문을 포함시킬 계획이다. 이것은 지난해 삼성전자와 삼성반도체통신을 통합하여 새로운 삼성전자를 출범시킨 이후 지속적으로 추진되어 온 일이다.

이러한 삼성의 계획은 제품특성에 따른 연구개발과 영업기능을 강화하여 시장경쟁력을 높여 세계적인 C&C 업체로 발돋움하기 위한 초석을 다지기 위한 것이 주된 목적이다.

—삼성종합기술원

갈륨비소 웨이퍼 개발

삼성종합기술원에서는 약 2년간의 개발기간을 거쳐 국내에서는 처음으로 LEC



(Liquid Encapsulated Czochralski :액체봉지인상) 공법에 의한 갈륨비소 (Ga-As) 단결정 및 웨이퍼 개발에 성공했다.

LEC란 현재 실용되고 있는 갈륨비소 단결정의 제조법으로 갈륨비소융액의 자유표면을 액체로 덮고 결정을 끌어 올리는 기법이다. 이 기법을 사용하여 만들면 결정의 고순도화와 대구경화가 가능하여 생산가의 대폭적 인하가 가능하게 된다.

또한 이 제품은 기존의 실리콘 반도체보다 정보처리 속도가 5배이상 빠르고 내고온 및 내방사성이 우수하다. 이 특성으로 인해 갈륨비소는 발광다이오드, 반도체, 레이저 등의 발광소자와 초고주파의 증폭 및 발진용 소자로 실용화되고 있으며, 실리콘 집적회로를 능가하는 고속집적회로에도 응용이 가능하다.

—한국과학기술원

한국과학원과 한국과학기술연구소로 분리 운영

한국과학기술원(KAIST)은 지난 6월 8일로 한국과학원(KAIST)과 한국과학기술연구소(KIST)로 분리되었다.

이로써 지난 81년 5월 통폐합된 후 8년여만에 원래의 운영체제로 복귀되는 것으로 과기처는 한국과학기술원의 학사부와 한국과학기술대학을 통폐합하여 한국과학원으로 이름을 변경할 방침이다. 한국과학기술원은 산하기관으로 유전공학센터, 시스템공학센터, 해양연구소 등과 그외 관련 부설 연구기관을 두어 앞으로 연구과제별 중심으로 재료과학, 응용과학, 기초 및 시스템공학 등 3개 연구단으로 개편하여 종합연구기관으로서의 역할을 수행하게 된다.

또한 특별연구로 연구과제당 40억~50억원 규모의 장기과제를 연 3~5건씩 선정, 지원할 계획이며, 아울러 약 10억달러 규모의 첨단 연구장비를 도입하여 연구시설의 현대화를 추진할 예정.

내년부터 단계적으로 이전하게 될 홍릉단지의 학사부도 대덕연구단지의 과기대와 통합시켜 학사, 석사, 박사로 이어지는 과학영재 교육을 담당하게 된다. 그리고 내년까지 교수대 학생비율을 1대 10정도로 줄여 교육의 질적인 향상과 학사조직, 교육과정의 개편 등 학사운영체제도 재정비할 계획이다.

새로 분리된 한국과학기술연구소의 초대소장은 한국과학기술원의 연구 본부장이었던 박원희 소장으로 내정되었으며, 한국과학원은 이상수 원장이 유임될 예정이다.

—삼성데이타시스템 (주)

골프장 관리용 전산시스템 개발

삼성데이터시스템 (대표 : 임경훈) 에서는 중앙개발 (주) 과 함께 골프장 업무전산화 전담팀을 구성하여 골프장 관리용 전산시스템을 개발하는데 성공하였다.

이번 상품화에 성공한 시스템의 특징은 클럽하우스를 중심으로 프로숍 (Pro Shop), 티하우스 (Tee House) 를 온라인으로 연결하여 현장에서 직접 전표를 입력하면 일일 영업현황을 쉽게 처리할 수 있을 뿐만 아니라 정확한 예약 및 신속한 요금관리를 할 수 있다.

따라서 본 시스템을 이용할경우에는 영업 및 회원관리, 급여관리, 인사관리, 재무관리, 캐디관리까지 골프장 경영에 필요한 전반적인 정보를 신속하게 제공하여 골프장 고객들에게 보다 폭넓은 서비스 제공이 가능하다.

이 제품에 대한 자세한 문의는 삼성데이타시스템 (주) 전화 759-1312 로 연락하면 된다.

—의료보험관리공단

의료보험 온라인 전산화 추진중

의료보험관리공단은 의료보험관리업무의 효율화를 위해 공무원 및 사립학교 교직원에 대한 의료보험 온라인 전산화를 추진중이다.

의료보험관리공단은 1차로 서울 순천향병원을 연결하여 온라인 시범운영중인데 환자 등에게 호응도가 높아서 앞으로 단계적으로 전국의료기관에 확대해 나갈 방침이다.

의료보험 전산화작업을실시함으로써 의료기관이 수시로 대상자의 자격과 진료비 지급일정 등을 즉시 조회할 수 있으며 의료보험카드 분실시에도 자격을 쉽게 확인할 수 있는 등 복잡한 과정이 불필요하게 되어 의료보험이용자의 병원 이용이 한층 편리해지게 된다.

반면 병원측은 자금업무효율을 높일

수 있다. 그러나 병원측의 인식재고와 다양한 기종간의 인터페이스문제 등으로 인하여 조기확산에는 다소 무리가 있을 것으로 전망된다.

—인하대학교

이주근 교수팀, SPD 램 세계 최초 개발

소형컴퓨터의 캐시기억소자와 중, 대형 컴퓨터의 주기억장치, 산업 및 통신기기 등의 기억장치에 널리 이용 가능한 SPD (Single Port Dual) 램이 인하대학교 전자과 이주근 교수팀에 의해 약 5년에 걸친 연구끝에 개발되었다.

이 SPD램은 S램과 D램을 하나의 소자에 집적시킨 것으로 기존의 D램과 S램의 데이터버스와 제어버스를 공유하는 구조로 되어 있고, 제어회로 역시 공유하여 집적도를 크게 향상시켜 다양한 규격의 SPD램을 개발할 수 있을 뿐 아니라 4M 및 16M로의 용량 확장이 용이하다.

제조공정 또한 D램과 비슷하여 가격도 저렴하다. 이처럼 SPD의 등장으로 기존 D램과 S램에 있던 가격과 속도면의 단점을 해결하게 되었다.

—(주) 콤텍시스템

대우증권에 논스톱 온라인망 구축

(주) 콤텍시스템 (대표:남석우)은 지난 5월말 대우증권(주)과 각종 데이터 통신망을 본점에서 효율적으로 운용 관리할 수 있는 망관리시스템(NMS: Network Management System)과 다기능 MUX 및 인텔리전트 모뎀 등 코덱(Codex) 종합 망시스템 공급 계약을 체결하였다.

코덱 종합 망시스템은 증권투자자들이 적시에 정보를 분석할 수 있도록 논스톱 (Non-stop) 망에 역점을 두었으며, 전송효율이 높은 STDM (통계적시분할 다중화장비)을 기반으로 하여 X.25 패킷스위칭 기능을 부가할 수 있고, 전체 망을 센터에 위치한 INS (Integrated Network Management System)에 의해 운용 관리할 수 있는 특징을 갖고 있다.

따라서 대우증권에서는 언제라도 고객이 필요한 정보를 실시간으로 제공받을 수 있으며, 자체지점 전산업무도 온라인 장애없이 지속적으로 처리할 수 있어 대고객 서비스의 질을 한층 높이게 되었다.

—한국데이타통신 (주)

팩스이미지 서비스용 정보통신 시스템 개발 계획

한국데이타통신 (주) (대표 : 이응효)은 광디스크에 각종 정보를 저장해 두었다가 이용자가 퍼스널컴퓨터로 검색하여 필요한 정보를 요구하면 전송해주는 '팩스이미지 서비스'용 소규모 시험시스템 개발에 착수할 계획이다.

이 소규모 시험시스템을 동사는 광디스크장비를 도입하여 광디스크에 화상정보를 입력, 화상 데이터베이스를 구축한 뒤 메인 컴퓨터를 통해 사용자의 단말기를 광디스크장치에 연결하는 방식으로 개발하여 10월경에 시험서비스에 나설 예정이다.

그리고 이를 착수하기 전에 입력 시간을 단축시키고 사진이나 그림 등과 함께 저장될 필요가 있는 정보를 구축하고 DNS (공중정보통신망)을 통해 일본의 ELNET (전자도서관 서비스) 를 국내시험 서비스할 계획도 추진중이다.

—KDC 상사

카드리더 판매 개방

KDC 상사는 전화기 산업기술연구조합 회원사인 7개업체에만 공급하여 오던 카드리더를 타 업체에까지 확대 공급키로 하였다. 이에 따라서 KDC 는 통신공업협동조합 산하의 업체들에게는 업체가 원할때에 이를 공급키로 하였다.

동사의 이러한 방침은 최근 카드식 공중전화기에 참여하려는 신규업체들이 카드리더 공급을 요구한 것과 KDC 와 연구조합간의 유착을 우려하는 여론이 높다는 것에 따른 것으로 여겨진다.

한편 KDC 상사는 카드리더 국산화 일정을 확정하여 카드리더 가격을 절감하는 방안을 마련하였으며, 기술 제휴사인 스위스 오텔카사로부터 국산카드리더의 성능을 평가받음에 따라 역수출하는 방안도 고려중이다.

—한국정보산업연합회

한국정보산업발전계획 수립

한국정보산업연합회 (대표 : 이용태)는 2000년대초 국내 정보화 사회상 실상과 더불어 좀더 종합적이고 체계적인 정보산업 육성을 위한 '한국정보산업발전계획' 수립을 구체화하였다.

정부의 정보산업 육성정책구축 및 체계적인 정보화 실현의 모델이 될 이 계획수립으로 우선 회원사 중심으로 관·산·학·연구계 대표와 중진인사 등 150명을 구성하여 '정보산업발전계획협의회' 를 발족하였다.

이 협의회는 정보산업분야 발전의 기본방향 및 정책화 구현방법을 설정할 정보기기, 반도체 및 부품, 정보처리, 정보통신, 정보화 촉진 등 5개 부문의 실무작업반을 편성하여 6월말까지 1차로 각 부문별 발전계획을 수시 검토하고, 9월중 공청회를 거쳐 10월말에 최종 마무리를 지어 관계당국에 건의할 예정이다.

그리고 이 장기계획 수립에 있어서 각 부문별로 현황을 파악, 여기서 발생되는 문제점의 해결 방안과 고도의 정보화 추진방향을 설정할 계획이다.

—왕컴퓨터 코리아 (주)

삼성물산과 전제품 국내판매 계약체결

왕컴퓨터 코리아 (주) (대표: 조선형)

• PC 레이다 •



는 지난 5월 26일 삼성물산(주)과 왕전제품에 대한 국내 판매 계약을 체결했다.

이번 계약을 통해 삼성물산에서 판매하게 될 제품은 분산처리 기능이 뛰어난 VS 5000/7000/10000 시리즈 등의 수퍼미니 컴퓨터시스템과 화상을 포함한 문서의 처리와 보관을 위한 VS 미니컴퓨터 기반의 통합화상 정보처리시스템(WIIS), 그리고 컴퓨터 키보드에 익숙하지 못한 경영자를 위해 PC에 손으로 쓴 문서와 음성정보를 입력하여 그 문서를 송수신할 수 있도록 한 Free-Style시스템 등이다.

삼성물산 정보기기 사업부는 이번 계약체결에 대해 "왕의 VS 시스템을 기반으로 한 수출서류관리 및 자동화 시스템을 자체적으로 개발하여 사용해 본 결과 시스템의 우수성과 사용의 편리함을 인식하게 되어, 이 시스템을 턴키베이스로 판매키 위해 이번 계약을 적극적으로 추진하였다."고 밝혔다.

왕컴퓨터는 삼성물산 이외에도 코오롱상사, 대우전자, 대성산업과 시스템 판매계약을 체결한 바 있다.

—한국컴퓨터연구조합

한글/한자 그래픽바이오스 모드 표준안 마련

한국컴퓨터연구조합에 의하면 금성사, 삼성전자, 대우전자, 현대전자, 삼보컴퓨터 등 국내 15개 컴퓨터업체들은 자사에서 생산하는 IBM PC 호환기종의 한글소프트웨어를 호환성있게 사용할 수 있도록 지난 5월 17일 한글/한자그래픽바이오스모드 표준안을 마련하여 상공부에 국가표준안으로 건의중에 있다고 한다.

이 표준안은 허큘리스, CGA, VGA 등 3가지 체계로 각 비디오그래픽보드에 맞게 제정되었으며, 최소한 호환성을 갖도록 하였다. 이 표준안이 확정될 경우, 각 PC간 한글소프트웨어의 호환성문제가 해결되어 중복적인 개발방지와 한글소프트웨어패키지의 데이터호환 등으로 국내 소프트웨어 시장이 활성화될 것으로 기대된다. 특히 컴퓨터 사용자의 활용폭이 넓어짐으로써 컴퓨터마인드 확산과 보급에도 많은 영향을 줄 것으로 전망된다.

IBM PC 호환기종에서 비디오 그래픽보드는 비디오 입·출력 모드값이 40H인 경우 그래픽 타입으로 문자가 처리되어 한글/한자가 16×16 도트로 표시되고, 화면해상도는 600×400으로 처리된다.

각 보드별 표준안을 살펴보면 허큘리스보드의 경우 비디오 입·출력 모드값이 7일때 텍스트형으로 한글/한자가 80자×25줄로 화면에 표시되고, CGA 보드의 경우 입·출력 모드값이 2 또는 3일때 흑백과 컬러로 한글/한자를 80자×25줄로 표시한다. VGA보드의 경우 허큘리스와 CGA보드 바이오스를 대부분 수용하고 모드값이 60 또는 70일때 16가지색으로 한글/한자를 표시하도록 권장하고 있다.

그외 영문과 한글/한자의 키보드 및 비디오 입출력 방법은 행정전산망 워크테이션에서 사용한 방법과 동일하게 규정하고 있으며, 상공부는 이 표준안을 재검토하여 국가표준 한글그래픽 처리방법으로 제정할 계획으로 전해졌다.

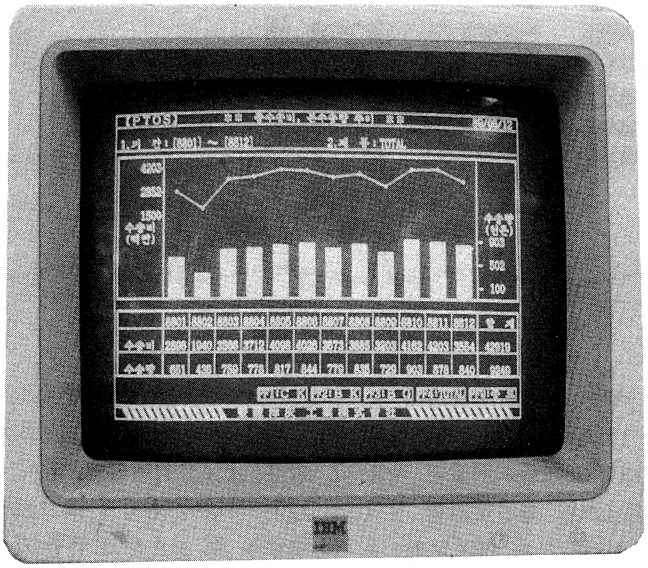
—쌍용컴퓨터

"제3회 한국 S/W공모전"서 종합물류계획시스템 대상 수상

쌍용컴퓨터(대표:최병항)의 '종합물류계획시스템'(TLPS:Total Logistics Planning System)과 범용 전문가 시스템 구축을 위한 도구인 'MORUS'가 '제3회 한국소프트웨어 공모전'에서 각각 대상과 장려상 수상작으로 선정되었다.

이번 공모전의 주최측인 한국정보처리전문가협회의 심사위원들은 프로그램 개발기법과 실 사용자의 사용효과, 도큐멘테이션에 대해 중점적으로 심사했다.

대상 수상작인 종합물류계획시스템은 생산공장의 지역적 분산, 물동량 및 재고량 증대, 제품 및 수송수단의 다양화, 빈번한 계획 변경에의 대응 등의 요인으로 복잡해지고 있는 유통체계속에서 최소의 물류비용으로 제품의 수급계획을 수립하고, 계획에 근거하여 실시된 물류시스템을 모니터링하면서 실적을 평가할 수 있도록 개발된 시스템이다.



그리고 이 시스템은 공장별 유통관련 자료(생산, 재고, 수요 등) 및 저장시설, 수송경로, 수송매체, 수송단가 등에 관한 '자료의 종합관리 기능'과 소비지 물량배정 최적화 모델, 생산 및 재고계획수립 모델, 제품수급계획수립모델 등과 같은 '종합물류 최적화모델 제시기능'을 보유하고 있다.

또한 영업, 유통, 생산 등 외부 시스템과의 인터페이스 및 추이 분석(톤당 이송비, 총수송량/수송비, 생산/수요/재고량 등) 또는 계획대 실적비교(안전재고, 수송경로간 이송량, 내수량, 수출량 등) 등의 통계분석 및 그래프 작성 기능과 월별 제품수급총괄표 등 40여종의 보고서작성 기능을 보유하고 있다.

—한일전산개발(주)
지방상호신용금고 업무전산화 관련 세미나 개최

한일전산개발(주) (대표 : 양재학) 은 지난 6월 9일 인천 올림프스호텔 대연회장에서 35개 상호신용금고 사장을 비롯한 70여명의 임직원이 참석한 가운데 상호신용금고 업무전산화 세미나를 개최했다.

이날 세미나는 3부로 나뉘어 진행되었다. 1부에서는 금융전산화 현황과 금융전산화시 고려사항에 대해, 2부에서는 상호신용금고 하드웨어와 소프트웨어에 대해, 3부에서는 SMPS (상호신용금고 종합금융시스템) 의 소개에 대해 진행되었다.

동사는 지방 상호신용금고 업무의 전산화를 위해 노력한 결과 지난 5월과 6월에 걸쳐 부산의 국민•흥업상호신용금고, 대구의 대한•아진상호신용금고, 울산의 영남상호신용금고, 대전의 대한상호신용금고 등 6개의 지방 상호신용금고에 각각 미니컴퓨터 HP-3000/925와 HP-M3000/XE를 이용한 SMTS를 일괄적으로 공급하기로 계약을 체결한데 이어 경인지역의 성남유린상호신용금고, 인천 한국상호신용금고, 의정부 동아상호신용금고, 안양상호신용금고 등 4개 상호신용금고에도 업무전산화를 추진하고 있다.

이번에 개발된 SMTS V4.1 프로그램은 상호신용금고 전업무 온라인 패키지로 온라인을 통한 효율적인 MIS (경영정보시스템) 를 구축함으로써 금고의 내외 전반에 관한 정보를 통합적으로 데이터베이스화하여 각 분야에서의 의사결정을 신속정확하게 처리하는 것이 특징이다.

—왕컴퓨터 코리아(주)
윈도우 소프트웨어 'ClearView' 발표

왕컴퓨터 코리아(주) (대표 : 조선형)는 마이크로소프트 윈도우(MS Windows)를 보다 편리하게 사용할 수 있도록 도와주는 ClearView PC소프트웨어를 개발했다.

ClearView는 Wang의 PC 200시리즈, IBM PC/AT, PC XT286과 그 호환 기종, IBM PS/2 시리즈 등에서 사용할 수 있는 윈도우 작업 환경을 관리하는 것으로 윈도우에서 수행되는 모든 소프트웨어를 지원한다. ClearView는 기존 윈도우의 기능에 다양한 기능이 추가된 그래픽 아이콘 메뉴 인터페이스와 Window Organizer 유틸리티 기능을 갖고 있다.

그래픽 아이콘 메뉴 인터페이스로는 텍스트 또는 아이콘 기반의 메뉴를 만들어 보다 쉽게 윈도우 하의 어플리케이션을 선택, 수행할 수 있다. 여러개의 어플리케이션도 하나의 메뉴로 선택되며, 윈도우 어플리케이션 이외의 어플리케이션도 같은 메뉴에서 액세스가 가능하여 기존의 MS-DOS Executive 화일명을 대체할 수 있다.

Window Organizer 유틸리티로는 어플리케이션이 수행되는 윈도우의 크기와 위치를 원하는대로 조절할 수 있고, 윈도우의 개폐, 크기의 확대 및 축소, 이동 등의 일반 기능들을 자동화 하여 보관할 수 있다.

ClearView는 PageMaker, Wang Bridgemaker, Microsoft Excel 등 윈도우하에서 수행되는 모든 어플리케이션과 통합된다. Wang Freestyle 시스템과 함께 사용시에는 ClearView 스크린을 하나의 PC스크린과 같이 전자종이로 변환시켜 손으로 쓴 글씨, 음성 등을 합성할 수 있다. PC

서클동정

• 고려대학교 •

대학 써클로는 최초로 BBS 개설

고려대학교 컴퓨터 클럽(KUCC)은 대학의 컴퓨터 연구와 컴퓨터 학습을 통한 친목도모를 위하여 1973년 5월에 창립되었다.

현재 회원이 100명에 이르고 있는 동 클럽은 매학기 초 신입생에 한하여 회원을 모집하고 있으며 회원에 가입하게 되면 2달간의 초급과정으로 컴퓨터 활용의 기본규칙과 손쉬운 활용법에 관한 교육을 받게 된다.

초급과정이 끝나게 되면 1~2명의 선배와 신입회원 5명 정도가 한팀을 이루어 그룹별로 파스칼과 C, 어셈블리 등의 컴퓨터언어와 유틸리티에 관한 연구를 통하여 프로그램 작성 및 컴퓨터 시스템의 이해능력을 배양하고 있다.

동 클럽은 매년 자체의 실력향상 및 평가를 위해 1회의 전시회를 개최하고 있으며 연세대학교 컴퓨터써클과 공동으로 매년 프로그램 경진대회를 실시하여 양교의 실력향상과 정보교환, 친목도모를 다지고 있다. 또한 학술지의 성격을 띠는 "오대호"를 계간지로 발간하여 회원들의 학술활동과 회원의 단합을 유도하고 있다.

동 클럽의 임원진은 회장을 중심으로 총무, 회계, 여학생부, 학술부, 섭외부, 문예부장으로 구성되어 클럽을 운영하고 있다.

동 클럽은 대학교 써클로는 최초로 지난 4월에 BBS를 개설하고 이 BBS를 통해 각 학교의 컴퓨터써클 및 컴퓨터 동호인과의 정보교환을 하고 있다. 이 BBS의 호스트 운용 프로그램은 RBBS이고 시스템은 IBM PC XT에 20MB 하드디스크로 되어 있다.

원승필 회장(토목 3)은 BBS를 통해 많은 소프트웨어의 교환이 이루어져 많은 사람이 폭넓게 소프트웨어를 제공받을 수 있도록 노력할 것이며 클럽 자체의 실력향상을 위해 팀간의 교육 강화에 박차를 가할 것이라고 한다.

동 클럽에 관한 문의사항은 전화번호 (02)924-7136 고려대학교 컴퓨터 클럽으로 하면 된다.

하계 수련회를 통해 미래 정보산업의 역군이 되기를 다짐하였다고



• 엠팔 •

엠팔 BBS 개통식 가져

엠팔은 지난 5월 25일 동대문구 이문동 엠팔 랩에서 BBS 개통식을 가졌다. 그동안 엠팔은 한글 에뮬레이터를 개발하고 랩탑용 한글개발, 한글 프로그램개발 등의 개발활동으로 축적된 경험과 지식을 바탕으로 엠팔내의 기술지원 그룹인 엠팔 Tech 그룹이 3개월간에 걸쳐 완성시킨 동 BBS 프로그램은 조합형 및 완성형 한글이 지원되며 통신 프로토콜은 X modem, Crc modem, Y modem, Batch ymodem, kermit, Z modem, Sealink, Ascii 등 8개가 지원된다.

동 BBS를 운영하고 있는 시스템은 32비트 CPU i80386을 사용하고 있고 주기억용량 2MB, 클럭속도 26.7MHz, 40MB와 80MB 용량의 하드디스크와 운영체제로는 유닉스 시스템V를 채택하고 있다.

BBS의 운영시간은 하루에 22시간으로 오후 1시부터 3시까지는 시스템 유저를 위한 휴식기간이므로 이때는 운

영이 정지된다. 현재 개통된 전화번호는 967-8022~4번이다. 동 BBS에 가입하고자 한다면 통신 파라미터를 7비트/even, 8비트/none으로 설정하고 전화를 건 뒤에 sonnim이라는 로그인으로 접속한다. 그리고 가입을 원하는 메시지를 남기면 추후에 우편으로 가입의 결정사항과 사용자 부호(ID)와 암호를 받게 된다.

BBS에 가입하게 되면 BBS의 제반 운영에 쓰일 회비로 학생은 5천원, 일반은 1만원, 법인은 2만원을 매달 납부해야 한다.

엠팔 BBS는 개설이후 메일 전달 기능과 프로그램(binary)의 업/다운로딩, 특정 관심그룹의 포럼난을 개설하고 데이터 베이스를 구축하여 유용한 서비스를 제공할 계획에 있다. 그리고 BBS의 확산을 위해 현재의 포트를 100개 정도 더 늘릴 것이며 시스템도 수퍼 미니급으로 향상시킬 것이라 한다.

동 BBS에 관한 문의사항은 전화번호 (02)336-4585 고연희씨 앞으로 하면 된다.

•하이콤 네트•

부산지역 컴퓨터 통신의 새로운 장을 마련

하이콤 네트는 부산지역의 컴퓨터 사용자들에게 보다 높은 수준의 정보를 빠르게 서비스하고 컴퓨터 통신에 대한 관심을 고조함과 동시에 그 수요를 충족시킨다는 목적을 가지고 1989년 3월에 개통된 BBS이다.

현재 회원이 80명에 이르고 있는 동 BBS는 C와 PASCAL 등의 컴퓨터언어, 패키지, 통신 등의 분과로 세분되어 있다. 매월 세째 일요일에 정기적인 모임을 갖고 있으며 이 모임에서는 각 소속 분과별로 세미나방식의 강연과 토론을 통하여 회원을 교육하고 있다.

동 BBS에 가입을 희망하는 사람은 모뎀이 있어야 하며 동 BBS의 목적에 찬성하는 사람으로 전화나 메일을 이용하여 메시지를 남기고 가입비 3만원(학생일 경우 1만원)과 BBS의 확충과 하이콤 네트의 기술적인 지원 그룹인 HUSG(Hicom User Support Group)의 운영비로 쓰일 회비를 매월 1만원씩 납부하면 된다.

호스트 운영 프로그램으로 "Wildcat"을 사용하고 있으며 시스템으로는 IBM-PC/AT에 130MB의 하드디스크를 갖추고 있다. 또한 비회원을 위하여는 65MB의 하드디스크를 설치하고 있다. 그리고 시스템 오퍼레이터는 김장호씨와 최호영씨가 맡고 있다.

조재순 회장(재민 한의원)은 앞으로 무선 패킷 통신을 이용한 무선 BBS를 운영할 계획이며 또한 현재의 독립된 BBS를 멀티라인으로 묶어 사용자의 편리를 도모할 예정이라 한다.

동 BBS에 관한 문의사항은 전화번호 (051)862-2532 정원준씨 앞으로 하면 된다.

•전국대학 전산학과 연합회•

제2회 전시회로 도약의 발판 구축

전국대학 전산학과 연합회(NCA)는 정보화 시대의 요구에 부응하여 전산 관련학과가 점차 증설되자 지난 80년 경인지역 전산학과간의 정보교류 및 단합을 위해 경인지구 연합회가 결성되고 82년에는 이러한 모임이 전국적으로 확대되면서 탄생하게 되었다.

동 연합회는 현재 72개 회원교의 95개학과가 가입되어 전체 회원은 3만여명에 이르고 있다. 동 연합회의 제반 운영은 운영위원회에서 담당하고 있는데 운영위원회는 연합회의 대표이자 모든 활동을 총괄하는 회장과 각 지부의 연락을 담당하는 총무, 연합회의 행사를 기획하는 기획부장과 재정지원의 사업을 담당하는 회계, 연합회의 학술교류 및 각종 학술지를 발간하는 일을 담당하는 학술 편집부, NCA의 전반적인 홍보활동을 담당하는 섭외부장으로 구성되어 있다. 운영위원회는 매주 일요일마다 모임을 갖고 행사 집행에 관한 토의를 하고 있다.

동 연합회의 활동사항으로는 회원의 전체적인 단합을 위한 체육대회와 실력향상을 위한 전시회, 회칙의 수정 및 회장의 선출과 연합회의 행사의 결정을 위한 정기총회가 매년 1회씩 실시되고 있다. 학술적인 활동으로는 전산 관련학과의 전반적인 시설 현황을 파악하고 학교와 학과의 소개, 진로 등의 자료를 모은 정보지를 발간하고 있다. 회원의 교육은 각 학교의 자체적인 교육을 통하여 단계별로 실시하도록 하고 있다.

동 연합회는 5개의 지구로 구성되어 있는데 서울·경인지구는 32개교, 강원지구는 4개교, 영남지구는 16개교, 호남지구는 12개교, 충청지구는 8개교가 지구별로 단체행사를 주관하고 있으며 각 지구에는 그 지구에 소속된 학교의 대표가 모여 활동하고 있다. 그리고 지구별 정보의 교환은 지구의 대표가 있는 학교에 설치된 모뎀을 통하여 하고 있다.

동 연합회의 89년 계획으로는 방학을 이용하여 각 지역별 고교생을 대상으로 한 컴퓨터 특강의 실시와 정보지인 전국 전산관련과 자료집 제2호와 학술지를 발간하는 것 그리고 8월 21일 부터 25일까지 서울의 롯데월드 7층에서 개최할 제2회 전시회가 있다.

또한 동 연합회는 8월에 있을 전시회를 앞두고 제2회 전국 대학생 S/W 공모전을 실시하고 있다. 대상은 전국 대학생에 한하며 응모 부분은 업무용, 교육용, 생활정보, 유틸리티, 게임외 기타의 부분으로 16비트 PC로 작성한 순수한 개인 창작품으로 작품설명서와 기종 및 주변기기에 관한 서류를 첨부하여 7월 31일까지 동 연합회에 제출하면 된다.

동 연합회에 대한 문의사항은 전화번호 (02)791-1285 전국대학 전산학과 연합회로 하면 된다.

국외소식

일, 카시오
AX머신 2기종을 판매

카시오계산기(주)는 CPU에 80286(클럭주파수 12MHz)을 탑재한 AX규격의 랩탑머신「AX-8000L」2모델과 CPU에 386SX(클럭주파수 16MHz)를 탑재한 데스크탑 머신「A-8000」2모델을 개발, 판매에 나섰다.

AX-8000L의 주요 특징은

① 3.5인치 FDD 2대를 내장한「AX-8000L/M01」과 3.5인치 FDD 1대, 20M바이트 HDD 1대를 내장한「AX-8000L/M02」를 마련
② 디스플레이에는 8계조 표시가 가능한 해상도 640×480 도트로 백라이트가 부착된 흑백액정을 채용
③ 메인메모리는 640K바이트를 표준으로 장착 최대 2.6M바이트까지 확장가능
④ 인터페이스는 RS-232C I/F 2개와 프린터I/F 1개, 외부FDD용 I/F 1개, 텐키패드용 I/F 1개를 각각 장착
⑤ 확장슬롯은 전용보드용으로 1개 장착

등이며 사이즈는 312(W)×360(D)×89(H)mm이다.

AX-8000의 주요 특징은

① 3.5인치 FDD 1대와 20M바이트 HDD 1대를 내장한「AX-8000/M20」과 3.5인치 FDD 1대, 40M바이트 HDD 1대를 내장한「AX-8000/M40」을 마련
② 메인메모리는 1M바이트를 표준으로 장착하고 최대 7.6M바이트까지 확장가능
③ 인터페이스는 RS-232C I/F 2개와 프린터I/F 1개, 외부FDD용 I/F 1개, 키보드용 I/F 1개, CRT용 I/F 1개를 각각 장착
④ 확장슬롯은 PC/AT 풀사이즈보드로 4개 장착

등이다. 본체사이즈는 381(W)×382(D)×155(H)mm이다.

일, 통산성
위성통신시스템 기술연구소 발족

일본은 관민공동출자로 고도의 통신기술확립을 목표로 하는 위성통신시스템 기술연구소를 발족하였다.

우정성과 통산성이 70%를 출자하고 민간기업으로는 히다찌제작소, 미쯔비시전기, 오끼전기공업 도시바 등 전기메이커를 중심으로 12사가 참가, 나머지 30%를 출자하였다.

이 기술연구소의 연구테마는 이동통신기술이나 위성회선의 유효활용기술 등이다.

애플사
윈도우 소프트웨어 재판 승소

미국의 애플사는 Microsoft사와 Hewlett-Packard사를 상대로 윈도우 표시타입의 소프트웨어 저작권을 둘러싼 소송에서 승소하였다.

샌프란시스코 지방재판소에서 승소판결이 난 이 소송은 미 Microsoft사의 Windows Ver 2.03이 애플사가 가지는 윈도우 표시화면에 관한 특허의 사용허가를 받지 않았다는 내용이었다.

이번 승소로 애플사는 아이콘의 저작권에 대해서도 거론할 태세였으나 미, Microsoft사와 HP사측은 이에 대해 '아이콘을 사용한 메뉴화면은 어디까지나 기초적인 아이디어이며 저작권보호의 대상이 될 수 없다'고 반론하고 있다.

일, 도시바
J-3100시리즈로 새모델 2종류 발표

(주)도시바는 랩탑형 퍼스널컴퓨터 J-3100 시리즈로「J-3100GT」2모델을 발표하였다. 이들은 용량20M바이트의 HDD와 3.5인치 FDD 1대를 탑재한「021A모델」과 3.5인치 FDD 2대를 내장한「002A모델」이다.

주된 특징은

① CPU로는 클럭주파수 12MHz의 80286 을 내장하여 종래의 GT에

비해서 처리속도를 약1.5배 향상시켰다.
② HDD의 평균액세스시간은 29ms이다.
③ 메인메모리는 1M바이트를 표준장비로 하고 최대 5바이트까지 확장가능
④ 메모리는 EMS규격에 준한다.
⑤ 시리얼 인터페이스(RS-232C) 2개 장착
⑥ 박형플라즈마 디스플레이 등의 채용에 의해 본체중량의 약10% 경감
⑦ OS는 MS-DOS Ver3.1를 표준장착
⑧ 옵션으로 수치연산코프로세서 80287을 장착

등이다.

마이크로소프트사
Quick C 2.0발표

미, 마이크로소프트사는 C프로그래머를 위하여 집적화된 개발환경을 고려한 Quick C 2.0을 새롭게 선보였다.

이 Quick C 2.0은 기존 Quick C 사용자들이 하나의 메모리모델로 인해 느꼈던 지원부족과 반드시 C프로그램내에 인라인 어셈블러를 부착해야 하는 불편함을 해소하였다. 즉 5개의 메모리 모델과 인라인 어셈블리 언어에 대한 지원이 가능하다.

뿐만 아니라 내장편집기와 디버거를 향상시켰다. 편집기는 마이크로소프트 C 5.1에서 지원된 편집기에 상당히 근접하고 있으며 Epsilon Text Editor와 유사한 기능을 갖고 있다. 디버거는 종전보다 사용이 쉽고, 디버깅 개정동안의 단계별 과정을 유지하는 history 기능을 갖고 있다.

또한 Quick C 2.0의 가장 큰 특징은 Quick C Advisor이다. 이는 온라인 도움체계기능으로 사용에 필요한 실질적인 설명이 없이도 이용이 가능하도록 유도해 주는 기능이다.

그리고 사용자가 자신의 프로그램을 조정하고자 할때 DOS를 로드하거나 Quick C를 빠져나가지 않고도 가능하다. 또 다양한 표현이 가능한 향상된 그래픽 라이브러리를 갖고 있다. 즉 bar, 열, 라인, 파라그래프와 차트생성을 위한 루틴을 갖고 있고, 폰트를 다양하게 선택할 수 있다.

레퍼런스나 사용자 매뉴얼 대신 툴키트와 인스톨가이드를 갖고 있다. 이 인스톨가이드는 전반부에서 다른 언어로 프로그램 해왔던 유저를 위해 C를 주제로 함수와 데이터 형태, 연산자 등 전범위를 다뤄 기존 사용하던 언어를 C에서 어떻게 표현하고 실행하는지를 유도하는 내용을 묘사하였다. 후반부는 경험있는 C프로그래머에게 도움이 되도록 고수준의 주제를 다루었다. 툴키트는 컴파일링, 링킹, 라이브러리 함수관리와 같이 좀더 기술적인 문제를 다루고 있다.

미, Novell사
미니급이상인 Netware386 발표

미, 샌프란시스코에 위치한 Novell사는 기능상으로는 미니급이상이면서 기존 미니급에 비해 저가격이고 속도 또한 더 빠른 Netware386을 발표했다.

이 시스템을 지원하는 소프트웨어는 마이크로컴퓨터급에서 미니컴퓨터급까지 실행이 가능한 것으로 동사측에서는 이 소프트웨어가 하드웨어 능력보다 훨씬 가치가 있다고 말한다.

앞으로 이 시스템의 구체적인 발표를 두단계에 걸쳐 계획하고 있다. 그 첫단계로 올 3/4분기내에 Netware Ver3.0을 발표할 예정이다. 이 Ver3.0은 주기억용량이 16MB이고, 300MB의 하드디스크를 지원하는 인텔사의 80386을 사용하였기 때문에 250대의 개인용 컴퓨터를 지원할 수 있다.

또한 이의 가장 큰 이점은 곧 발표될 80486칩과의 연결까지도 예상해서 디자인하였다는 점이다.

두번째 단계로 내년 이후에 발표될 예정인 Ver3.1은 송신제어 프로토콜/내부연결 프로토콜인 X.25 유닉스와 Network File System사의 화일전송 프로토콜인 선마이크로시스템을 위한 제공을 구체화할 계획으로 밝혀졌다.

이번에 발표된 Netware386의 특징은 다음과 같다.

- Netware 프로그래머 작업영역을 이루기 위해 개발환경을 분류했다.
- 안정성 향상—패스워드를 이용하도록 되어 있고, 화일과 디렉토리 보관표시를 보장
- 보드 추가—IBM의 MCA(Micro Chanel Architecture)를 위한 32비트 어댑터, 강화된 디스크 보조프로세서 보드, 디스크 I/O를 집적화하여 시스템장애 최소화.
- 설치용이—기능이 향상되긴 했어도 설치하는데 소요되는 시간은 15분이면 가능하다.

동사는 앞으로도 IBM시스템 네트워크 구조분류서비스인 LU6.2와 X.400에 데이터 능력, 데이터베이스 서비스, 편의성 등이 반영된 전통적인 MIS(경영정보시스템) 관련으로 보강할 계획이다. PC

프로시듀어 화일의 이용

이기성 / 장왕교재연구원 상무

dBASE Ⅲ PLUS에서는 편리한 표준 제어구조들, 즉 DO WHILE~ENDDO, IF~ENDIF, IIF( ), DO CASE~ENDCASE 등을 구비하여 사용자들로 하여금 효율적인 프로그램의 논리구성을 할 수 있도록 돕고 있다.

dBASE에는 이 외에도 더욱 능률적으로 프로그램을 작성할 수 있도록 하는 명령들과 기능들이 있는데 그중의 하나를 든다면 프로시듀어(Procedure)의 이용을 들 수 있다.

보통의 사용자들은 쉽게 지나쳐 버리기 쉬운 기능인데 알고보면 이런 종류의 기능(Procedure, Public 변수를 은폐시키기 위한 Private 등)은 프로그램의 흐름과 메모리 변수처리에 큰 도움을 줄 수 있는 고급기능에 속하는 것 들이다.

일반적으로 프로그램을 설계하다 보면 특정 모듈을 반복적으로 사용하게 되는데, 이때 그 모듈을 실행시킬 때마다 DO명령으로 해당 화일을 연다는 것은 다 아는 사실이다. 이 경우, 해당 모듈이 별개의 프로그램으로 존재하므로 매번 읽어들일 때마다 전체 디스크 영역에서 찾은 후 열어야 하는 단점을 감수해야만 하는 것이다.

프로시듀어 화일을 이용하면 이러한 디스크 액세스 시간과 지연을 줄일 수 있을 뿐만 아니라 하나의 프로시듀어 화일에 최고 32개까지의 procedure 루틴을 둘 수 있어 대부분 대기상태에 있다가 한번 메모리로 읽혀들여 빠른 속도로 메시지를 화면에 나타내 주는 도움말 화일 등에 효과적으로 이용할 수 있다.

프로시듀어 화일을 이용하려면 먼저 일련의 기능을 갖고 있는 procedure의 집합인 프로시듀어 화일을 만들어야 하는데 여기서 procedure 루틴과 프로시듀어 화일의 구분에 혼동을 해서는 안된다(그림 1 참조).

그림 1을 보면, 프로시듀어 화일 INFO.PRG는 특정한 기능들을 갖고 있는 proce1, proce2, proce3를 포함하고 있음을 볼 수 있다.

메인 프로그램에서 이 프로시듀어 화일을 이용할 경우가 생기면 우선 프로그램의 서두 부분에 이 프로시듀어 화일을 여는 SET PROCEDURE TO INFO.PRG를 포함한 후 원하는 기능별로 DO proce1, DO proce2,… 등을 기술해 주면 된다.

```
set procedure to INFO.PRG
    ∫
do proce1
    ∫
do proce2
    ∫
do proce3
    ⋮
```

실제 프로시듀어 화일을 만들기 위해서는 각 프로시듀어 이름을 procedure와 함께 지정하고 끝에는 기타 다른 모듈 프로그램과 같이 RETURN문이 있어야 한다.

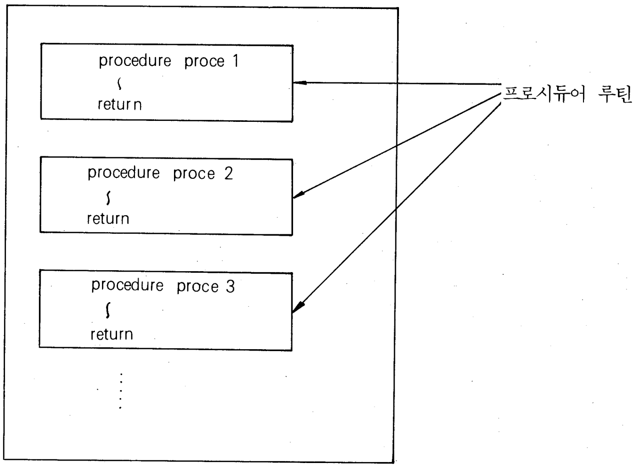
```
procedure proce1
  ⋮
return

procedure proce2
  ⋮
return
```

또 한가지 프로시듀어 화일은 한번에 하나의 화일만을 열 수 있기 때문에 새로운 화일을 열기 위해서는 다시 SET

〔그림 1〕 프로시듀어 화일의 구성

프로시듀어 화일 : INFO.PRG



PROCEDURE TO문을 이용해 새로운 화일이름을 지정해 주어야 한다.(이 경우 dBASE는 기존의 화일을 자동적으로 닫아준다.)

그럼 실제 예제를 가지고 프로시듀어의 이용법을 익혀보자. 리스트 1과 리스트 2의 프로그램을 modi comm명령으로 작성하고 나서 'do main'으로 실행을 시켜 본다. 여기서도 살펴볼 수 있듯이 일반적인 방법으로 do main.prg에서 다시 do sub1.prg를 부르고 sub1에서 한번 더 do subrtn.prg를 부르면 main.prg에서 사용하던 변수는 subrtn1.prg에서 기억을 못하는 수가 있다. procedure를 이용하면 이런 문제를 방지할 수 있으며 속도 또한 do ~.prg하는 것보다 set procedure to subrtn으로 지정한 다음, subrtn.prg안에 있는 procedure 루틴을 부르는 것이 더 빠르게 된다.

참고로, 리스트 3과 리스트 4에 변수의 값을 계산해 주는 procedure 의 예를 보였으므로 참고하기 바란다.

〔리스트 1〕 main.prg

```
*main.prg
*by leeks
set procedure to subrtn
clear
? " 지금 main.prg 입니다. "
wait "press <rtn> for testing subrtn-1" to s1
* subrtn.prg 내의 sub1 procedure를 부름
do sub1
?
? "=============================="
? " MAIN .PRG로 다시 오셨읍니다. "
? chr(7)
wait "press <rtn> for testing subrtn-2" to s2
*subrtn.prg내의 sub2 procedure를 부름.
do sub2
?
? "Welcome again returing from sub2-procedure!!!"
? "=================================="
? "MAIN으로 또다시 오신걸 환영합니다. "
? "=================================="
close procedure
*eof: main.prg
```

〔리스트 2〕 프로시듀어 화일 (subrtn.prg)

```
*subrtn.prg
*calling from main.prg
*by leeks
####################
procedure sub1
####################
? "This is sub1-proc "
dir
wait "press <rtn> for return to main2-pg. " to su1
return
###################
procedure sub2
###################
? "This is sub2-proc"
? date()
dir *.*
? "---------------------------------------------"
wait "press <rtn> again return to main-pg ." to su2
return
###
*eof : subrtn.prg
```

〔리스트 3〕 sijak. prg

```
*sijak.prg
*by leeks
*on 4322-5-10
set procedure to lksubrtn
clear
? "******************************************"
? "      시작 프로그램입니다.  "
? "******************************************"
a = 1
? "a = "
?? a
b = 1000
? "b = "
?? b
wait "press <rtn> for testing subrtn-1" to s1
###
do sub10
###
?
? "---------------------------------------------"
? " sub10 procedure에서 SIJAK.PRG로 다시 오셨읍니다."
? "---------------------------------------------"
? " a = "
?? a
? "b = "
?? b
```

```
? chr(7)
wait "press <rtn> for testing subrtn-2" to s2
***
do sub20
***
?
? "============================================"
? "SIJAK.PRG로 또다시 오신걸 환영합니다. "
? "============================================"
close procedure
*eof : sijak.prg
```

〔리스트 4〕 LK서브루틴(lksubrtn.prg)

```
*lksubrtn.prg
*by leeks
***************
procedure sub3
***************
 ?"This is sub3-proc"
 ?"-----------------"
 ?"Tested by leeks."
 wait "press <rtn> for return to Main.PG." to su3
return
***************
procedure sub10
***************
?"지금은 sub10 procedure 입니다. "
?"-------------------------------"
?"a * 300 = 300. changed from 1 to 300"
a = a * 300
?"a = "
?? a
?"b = "
?? b
wait "press <rtn> to sijak.prg ... " to sul
return
***************
procedure sub20
***************
?"----------------------------"
?"sub20 procedure입니다. "
?"----------------------------"
? date()
?" a = "
?? a
?" b = "
?? b
wait "<rtn> for returning " to sul
?"......................................"
return
***************
*eof : lksubrtn.prg
```

SEM-DRAW에 이미지 저장기능을

박세희/중앙대 셈틀연구회

지난 5월호에 소개한 SEM_DRAW는 그래픽 에디터로 활용하기에는 여러가지 부족한 기능들이 많았다.
따라서 여기에다 효과적으로 이용하기에 반드시 필요하다고 할 수 있는 몇가지 기능들과 몇군데 보완하여야 할 부분들을 수정해 보았다.
그래픽 이미지를 저장하고 복구하는 기능과 5가지의 글자 폰트를 가지고 크기를 마음대로 조절하면서 그림을 함께 넣을 수 있는 기능 등 몇가지 추가된 부분들은 SEM_DRAW의 기능을 한층 더 강화시켜 줄 것이다.

필자는 지난 5월호의 '퍼스널컴퓨터'에 '마우스를 이용한 그래픽에디터 SEM_DRAW'라는 글을 적은 적이 있다. 원래 이 글은 요즘 인기를 얻고 있는 마우스를 가지고 활용하는 한 예를 보이기 위한 것이었기 때문에 그래픽 이미지를 저장하거나 복구시키는 기능 등 그래픽 프로그램이 갖추어야 할 대부분의 루틴들이 생략되어 있다. 그러나 그래픽 프로그램이 애써 그린 그림을 저장하지 못한다면 결코 좋은 프로그램이 될 수 없다고 생각되기에 이번 짧은 지면을 얻어 그래픽 이미지를 저장하고 복구하는 함수와 5가지의 글자 폰트를 가지고 크기를 마음대로 조절하면서 그림과 함께 넣을 수 있는 함수를 소개하고자 한다. (퍼스널컴퓨터 5월호를 보면 터보 파스칼에서 쿼트로의 글자 폰트를 이용하는 예가 나와 있는데 필자의 프로그램에서도 이들 모두의 폰트를 이용할 수 있다. 이글에서는 실제 이용에 목적이 있는 것이 아니라 폰트를 이용하는 함수를 소개하는 것이 목적이므로 5개만을 사용했다. 관심있는 독자들은 직접 원하는 폰트를 이용해 보기 바란다)

그리고 필자가 지난번에 소개한 SEM_DRAW를 검토해 보니 비효율적인 부분을 몇군데 발견할 수가 있었기에 이 지면을 빌어 수정을 하고자 한다. 이런 수정, 보완작업은 앞으로도 독자들이 계속 해주길 바란다.

SEM_DRAW에서 수정할 부분

우선 처음 부분의 Line함수를 다음과 같이 구성한다. 이것은 선을 그려줄 때 정상적으로 되지 않고 한쪽으로 치우치는 것을 막기 위한 것이다.

```
void Line(int sx,.....)
{
 ....
 d_x=abs(d_x)+1;
 d_y=abs(d_y)+1;
 ....
 if (d_x>d_y) distance = d_x;
 else distance = d_y;
 ....
}
```

그리고, CloseWindow함수 부분을 다음과 같이 구성한다. 이는 사용하는 화면의 범위를 안정시키기 위한 것이다.

```
int CloseWindow(void)
{
  ....
  putimage(....);
  if (win_pointer==0) {
     xloc=-2;yloc=-2;wloc=721;hloc=349;
  }
  else {
     xloc= win_xy...
     ....
  }
  setviewport(....);
  ...
}
```

마지막으로, FILL_BOX에서

```
case FILL_BOX :
  ....
  if (fun==BOX)
```

```
   ...
 else {
   ...
   bar(xs,ys-11,c,d-11);
   ...
```

여기서 bar() 함수의 인자값을 수정하였다.

SEM_DRAW에 추가되는 부분

SEM_DRAW의 기능을 보강시키기 전에 다음의 내용을 첨가한다. 각종 수치변경과 메뉴 수정(화면 1 참조), 함수 추가에 대한 부분이니 꼭 필요하다.

```
 enum function {
     PENCIL    = 0,
     ......
     LETTER    = 6,    /* 추가 한 함수 */
     ERASE     = 7,    /* 추가한 함수 */
     CLEAR     = 8,
     LOAD      = 9,    /* 추  가 */
     SAVE      = 10,   /* 추  가 */
     EXIT      = 11
 };
 char *menu[] = {
     " Pencil   ",
     .....
     " Letter   ",     /* 추  가 */
     " Erase    ",
     " Clear    ",
     " Load Pict",     /* 추  가 */
     " Save Pict",     /* 추  가 */
     " Exit     "
 }
/* 추가되는 메뉴 데이타 */
 char *s_fonts[] = {
     " Default  ",
     " Triplex  ",
     " Small    ",
     " Sansserif",
     " Gothic   "
 };
 int s_font=0;           /*  현재사용중인 글자체를 기억하는 변수 */
 ....
 void system_set(void)
 {
    ....
    bar(0,0,87,7);
    size=imagesize(0,0,87,7);          /* pull down menu에서 /*
    ....                               /* 사각형의 커서크기를 줄이다. */
    getimage(0,0,87,7,menu_buffer);
    ....
 }
```

```
....
void Draw_Main(void)
{
  ....
  gprintf(6,1,"Pencil    ");          /* 이중 따옴표 안에 */
  gprintf(18,1,"Pattern   ");         /*   글자의 갯수가   */
  gprintf(28,1,"Solid     ");         /* 모두 10개가 되게 한다. */
  gprintf(40,1,"Color     ");
  gprintf(50,1,"White     ");
  gprintf(60,1,"Font ");
  gprintf(65,1,"Default   ");
}
.....
int select_drawing(void)
{
   ....
   OpenWindow(1,2,11,12);   /* 수치 변경 이필요한 부분들 */
   for (i=0,i<12,i++)
   ....
   while(b&1) {
     ...
     if (y>11) y=11;
   ....
   gprintf(6,1,"%s",menu[y]);
   ....
}
int select_pattern(void)
{
   ....
   OpenWindow(18,2,11,12);
   ....
   CloseWindow();
   gprintf(28, ... );
   ....
}
int select_color(void)
{
   ....
   OpenWindow(40,2,11,2);
   ....
   CloseWindow();
   gprintf(50, ... );
   ....
}
......
main()
{
   ....
   int lett[80]=" ";                 /* 새로운 변수가 추가된다. */
   ....                              /* 이것은 그림에 나타낼    */
   do {                              /*  글자를 보관한다.       */
     ...
     if (b&2) {                      /* 오른쪽 버턴이 눌러졌나 검사한다. */
        get_letter(&lett[0]);        /* 눌러졌을경우 글자를 받아온다. */
```

```
        fun=PENCIL;
        patt=1;
    }
    while {
        ....
        if (d<10) {
          if (c<48)                    /* 150을 48로 고친다. */
            fun = select_drawing();
            if (fun==LOAD) {    /* 새로 추가되는 기능을 실행하는 */
              hidemouse();      /* 부분들이다. */
              load();
            }
            else if (fun==SAVE) {
              hidemouse();
              setfillstyle(SOLID_FILL,0);
              bar(0,0,719,10);
              save();
            }
            if (fun==SAVE || fun==LOAD) {

/* 이부분은  text상태로 들어갔을때 잊어버리는 graphic상태를
   복구시켜 준다.  */
              setfillstyle(SOLID_FILL,0);
              bar(0,0,719,10);
              Draw_Main();
              fun=PENCIL;
              patt=1;
              pokeb(0x0040,0x0049,6);
              a=0;Mouse(&a,&b,&c,&d,&e);
              set_cursor();
              showmouse();
              p_color=1;
            }
            else
            if (c>144 && c<224)          /* 몇개의 숫자를 고친다. */
              ....
            else
            if (c>320 && c<400)
              ....
            else
            if (c>480 && c<520) /* 글자체를 바꾸는 함수를 부른다. */
              select_font();
        } else
            ....
            case BOX :
            case FILL_BOX :
            case LETTER :
              .....
              if {
                ...
              }
              ...
              if (fun==LETTER)
                put_letters(xs,ys,xe,ye,&lett[0]);
              else {
                if (fun==BOX)
                ...
              }
          .....
      ....
    }
```

리스트 1을 보면 추가시킬 새로운 함수들이 나온다. 이 중에서 hidemouse( )와 showmouse( )는 커서 숨기기와 다시 보이게 하는 부분을 별도로 분리하여 함수로 만들었다. 그러나 이글에서 그 수정위치는 생략하나, Turbo C의 에디터 상태에서 수정하는 방법을 설명하고자 한다. 다음 순서대로 키를 입력한다.

① CTRL-HOME
② CTRL-Q
③ CTRL-A

이것은 필요한 문장을 찾아서 대치시키는 부분을 부른다. 제대로 동작하면 화면위에 'Find:'란 메시지가 나오고 여기에 'a=1; ↵ '을 입력하면 이것을 어떻게 수정할 것인가를 묻는 'Replace with:'라는 메시지가 나온다. 이 상태에서 's-howmouse ( ); ↵ '를 입력하면 'Option'이 나오는데 그냥 ' ↵ '를 친다. 그러면 커서는 'a=1;'에 있고 화면상단에 수정할 것인가를 묻는데, 여기서 커서를 보이게 하는 부분(a=1;, Mouse(&a, &b, &c, &d, &e);)이면 'y'를 누른다. 이 작업이 끝나면 이제 불필요해진 'Mouse(&a, &b, &c, &d, &e);'를 'CTRL-Y'로 제거시킨다.

④ CTRL-L

이것은 지금 커서가 있는 위치 이후에 또 그 문자열이 있는지를 찾아본다. 그리고 이 작업(1번에서 4번까지)을 화면상단에 <ESC>키를 누르라는 메시지가 나올 때까지 반복한다. 그 다음 위와 같은 방법으로 'hidemouse( )'도 바꾸어 넣는다.

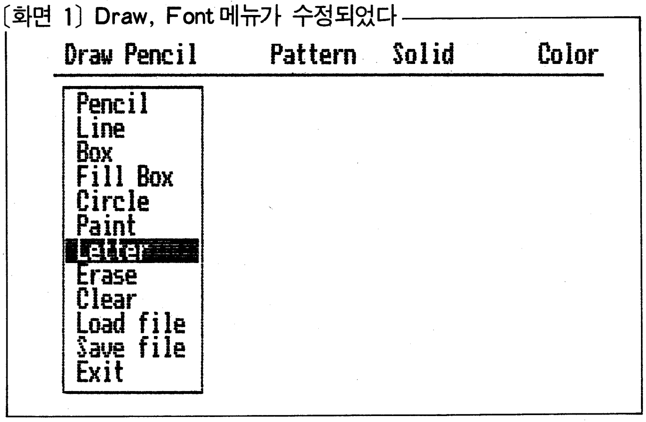
그리고 위와 동일한 방법으로 X, Y 좌표 표시부분을 다음과 같이 고친다.

```
gprintf(79,1,"%5d",c);
gprintf(85,1,"%5d",d);
```

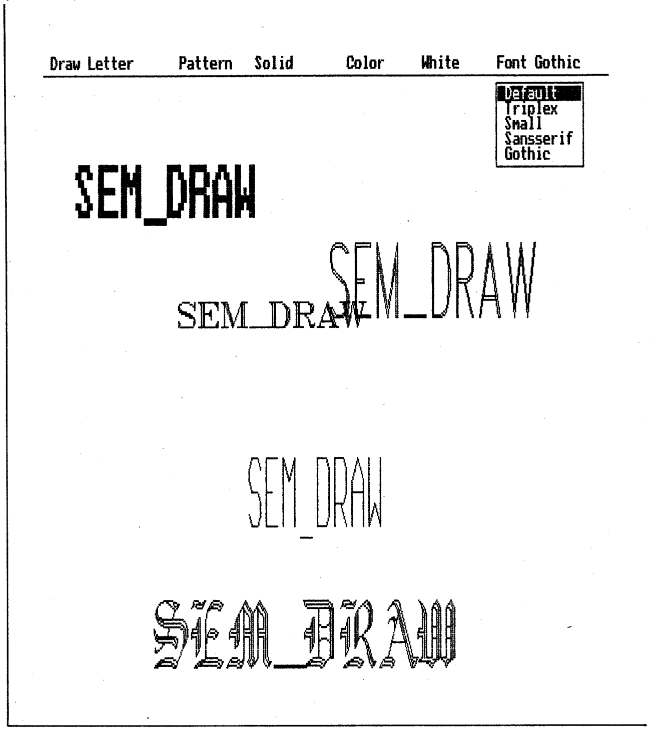
즉 70은 79로, 80은 85로 수정한다.

hidemouse(), showmouse()가 들어가야 할 위치는 select_drawing()함수 앞에 두면 된다. 그밖에 리스트 1에 나오는 함수들은 main()함수 앞에 위치하면 된다. 다음은 여기에 첨가되는 함수들에 대해서 설명하겠다.

〔화면 1〕 Draw, Font 메뉴가 수정되었다



〔화면 2〕 마우스를 이용해서 폰트의 크기를 조정할 수 있다.



함수 설명

load()와 save()는 그 성격이 비슷하게 되어 있으며 저장되는 형태는 2진 코드로 기억되게 된다. 그리고 저장하거나 로드할 때 원하는 화일의 이름을 묻는 과정에서는 텍스트 모드로 돌아가기 때문에 그림의 정보는 메인 메모리에 일시 기억시킨 후 디스켓에 들어가거나 나온다. beep( )함수는 전해주는 인자의 값에 따라 '삐'소리를 길게 또는 짧게 낸다.

- **int select_font(void)**

5가지의 글자체를 선택하는 함수이며, 선택 방법은 지금까지의 선택하는 함수와 동일하다.

- **void get_letter(char *sent)**

화면에 나올 글자들을 받아들이는 곳으로 텍스트 모드상에서 이루어진다. 마우스의 오른쪽 버튼을 누르면 글자를 입력할 수 있는 단계로 들어간다. 단, 여기서 스페이스 바를 누르면 그 이후의 글자가 무시되어 버리기 때문에 공백을 입력할 때에는 스페이스 바 대신에 아래줄을 입력시키기 바란다.

- **void put_letters(int sx, int sy, int ex, int ey, char *sent)**

위의 수정하는 부분에서 BOX, FILL_BOX, LETTER는 함께 묶어 놓았는데 이렇게 한 이유는 글자를 찍을 때 그 문장의 크기를 사용자가 지정한 사각형 안에 들어갈 수 있도록 조정하기 위해서이다. 사각형의 크기가 적당할 때 왼쪽버튼을 떼면 select_font()에서 지정한 폰트대로 화면에 글자가 나타나게 된다(화면 2 참조).

본지 5월호에 SEM_DRAW가 실린 후 그림을 저장시키는 프로그램으로 실었으면 좋았을텐데 하는 생각이 들었다. 그리고 어떤 독자는 일부러 뺀 것이 아니냐고 문의해 오는 분도 있었기에, 비록 특이한 것은 없지만 본란을 빌어서 글을 썼다. 수정할 것이 좀 많은 것 같다. 수정작업이 귀찮을지라도 기능을 하나더 얻은 셈치고 참아주기를 바란다. 그리고 글자 폰트처리는 load(), save()만으로는 부족한 것 같아서 함께 첨가하였다. 아무쪼록 이번에 실린 내용이 독자들에게 조금이라도 도움이 되었으면 한다. 그리고 나중에 알게 된 사실이지만 '퍼스널컴퓨터' 중간중간에 들어있는 삽화가 필자의 프로그램, SEM_DRAW로 그려지고 있다는 것을 알고 얼마나 기뻤는지 모른다. 이 기회를 빌어 퍼스널컴퓨터에게 고맙다는 말을 전하고 싶다. PC

〔리스트 1〕 새로 추가되는 함수 리스트

```
void showmouse(void)
{
   int a,b,c,d,e;
   a=1;
   Mouse(&a,&b,&c,&d,&e);
}
void hidemouse(void)
{
   int a,b,c,d,e;
   a=2;
   Mouse(&a,&b,&c,&d,&e);
}

int select_font(void)
```

```
{
   int a,b,c,d,e;
   int y=0,oy=0,ypos;
   int i;

   hidemouse();

   OpenWindow(60,2,11,5);
   for (i=0;i<5;i++)
      gprintf(1,i+1,"%s",s_fonts[i]);

   a=3;  /* call #3 to find whether left button
                                    is down */

   Mouse(&a,&b,&c,&d,&e);
   ypos = y*8;
   putimage(0,ypos,menu_buffer,XOR_PUT);

   while (b&1) {
      y=d/20;
      if (y>4) y=4;
      if ( y != oy ) {
         ypos = oy*8;
         putimage(0,ypos,menu_buffer,XOR_PUT);
         ypos = y*8;
         putimage(0,ypos,menu_buffer,XOR_PUT);
         oy=y;
      }
      a=3;
      Mouse(&a,&b,&c,&d,&e);
   }
   CloseWindow();
   gprintf(65,1,"%s",s_fonts[y]);

   showmouse();.

   return y;
}
void beep(int le)
{
   sound(1000);
   delay(le);
   nosound();
}
void get_letter(char *sent)
{
   int mode;
   int size;
   int a,b,c,d,e;
   void *buffer;

   hidemouse();
   size=imagesize(0,11,719,347);
    mode=getgraphmode();
    buffer=malloc(size);
   getimage(0,11,719,347,buffer);
   restorecrtmode();
   clrscr();
   printf("\n\n\n\n sentence to print...\n");
   printf("-----> no space bar allowed!
                   (use under bar)\n\n");
   scanf("%s",sent);

   setgraphmode(mode);
   putimage(0,11,buffer,COPY_PUT);
   free(buffer);
   beep(50);
   setfillstyle(SOLID_FILL,0);
   bar(0,0,719,10);
   Draw_Main();
   pokeb(0x0040,0x0049,6);
   a=0;Mouse(&a,&b,&c,&d,&e);
   set_cursor();
   showmouse();
   p_color=1;
   s_font=0;
}
void put_letters(int sx,int sy,int ex,int ey,
                              char *sent)
{
   int chx,chy,ccx,ccy;

   setusercharsize(1,1,1,1);
   settextstyle(s_font,HORIZ_DIR,USER_CHAR_SIZE);
   ccx=textwidth(sent);
   ccy=textheight(sent);
   chx=abs(sx-ex)+1;
   chy=abs(sy-ey)+1;
   setusercharsize(chx,ccx,chy,ccy);
   settextstyle(s_font,HORIZ_DIR,USER_CHAR_SIZE);
   if (s_font==0)
      settextstyle(s_font,HORIZ_DIR,chx/ccx+1);
   outtextxy(sx,sy,sent);

   settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
}

void load(void)
{
   void *buffer,*loaded;
   char name[20];
   FILE *fp;
   int a,b,c,d,e;
   long curpos,length;
```

```
  unsigned size;
  int mode;

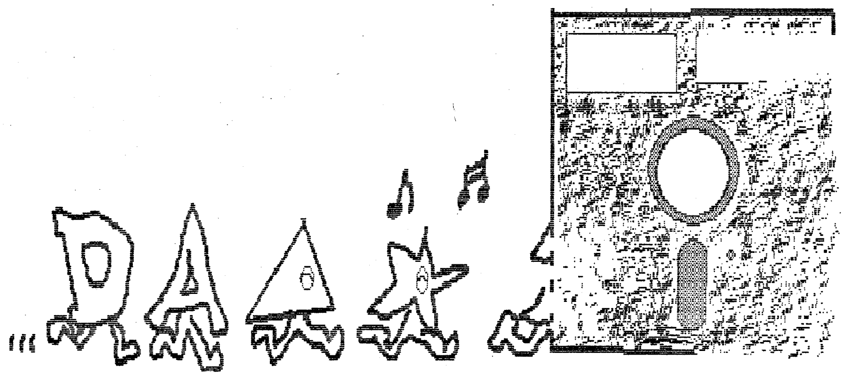
  size=imagesize(0,11,719,347);
  buffer=malloc(size);
  getimage(0,11,719,347,buffer);
  mode=getgraphmode();
  restorecrtmode();
  clrscr();
  printf("\n\n\n\n file name to load...\n");
  scanf("%s",&name[0]);

  if ((fp=fopen(&name[0],"rb"))==NULL) {
     printf("\nload failed.");
     printf("\npress left button");
     beep(100);
     do {
        a=3;
        Mouse(&a,&b,&c,&d,&e);
     } while (b!=1);
     setgraphmode(mode);
     putimage(0,11,buffer,COPY_PUT);
     free(buffer);
     return;
  }
  free(buffer);
  curpos = ftell(fp);
  fseek(fp, 0L, SEEK_END);
  length = ftell(fp);
  fseek(fp, curpos, SEEK_SET);
  loaded = malloc(length);
  fread( loaded,sizeof(char), length, fp);
  fclose(fp);
  setgraphmode(mode);
  putimage(0,11,loaded,COPY_PUT);
  free(loaded);
  beep(50);
}
void save(void)
{
  void *buffer;
  FILE *fp;
  char name[20];
  int a,b,c,d,e;
  long curpos,length;
  unsigned size;
  int mode;

  size=imagesize(0,11,719,347);
  buffer=malloc(size);
  getimage(0,11,719,347,buffer);

  mode=getgraphmode();
  restorecrtmode();
  clrscr();
  printf("\n\n\n\n file name to save...\n");
  scanf("%s",&name[0]);

  if ((fp = fopen(&name[0], "w+b"))==NULL ) {
     printf("\n\nSave failed..");
     printf("\npress left button");
     beep(100);
     do {
        a=3;
        Mouse(&a,&b,&c,&d,&e);
     } while (b!=1);
     setgraphmode(mode);
     putimage(0,11,buffer,COPY_PUT);
     free(buffer);
     return;
  }
  fwrite(buffer, sizeof(char),size,fp);
  fclose(fp);
  beep(50);
  setgraphmode(mode);
  putimage(0,11,buffer,COPY_PUT);
  free(buffer);
}
```



NKP용 도스 유틸리티

구준회/ 고대 컴퓨터클럽

NKP상에서 윈도우를 구현할 수 있도록 한 도스 유틸리티는 팝업메뉴 방식으로 메뉴를 선택하며 복수개의 화일을 선택할 수 있는 화일 선택기능이 있고 화일의 이름변경, 삭제, 인쇄, 도스쉘과 디렉토리 그리고 드라이브의 변경 기능이 있어 화일 조작의 편리함을 제공하고 있다.

지난 호에 NKP용으로 윈도우 함수들을 소개했었다. 사실, 이러한 함수들은 그 자체 만으로는 별로 가치가 없으며 다른 응용(Application) 프로그램들을 만들기 위해서 사용되어야 제 빛을 발휘한다. 이번 호에는 지난 호에 소개되었던 윈도우 함수들을 사용해서 **리스트 1**과 같은 NKP상에서 동작하는 간단한 도스 유틸리티를 만들어 보았다. (윈도우 함수에 대한 이야기는 본지 5월호를 참조하기 바란다.)

사실 PCTOOLS과 비슷한 기능을 될 수 있는 한 많이 첨가하려고 했으나 필자가 게으른 탓에 많은 기능을 첨가하지는 못했다. 앞으로 실력있는 사람들이 솜씨를 발휘하여 유용한 기능을 지닌 유틸리티가 많이 나타나기를 기대한다.

프로그램 구조

본 프로그램에 나타난 각 함수중 몇가지 중요한 함수들에 대해 자세히 알아보면 다음과 같다. 참고로 본프로그램은 터보 C 버전2.0에서 작성되었다.

프로그램의 전체적인 구성을 트리 구조로 보면 **그림 1**과 같다(리스트 1 참조).

위와 같은 구조에서 필자는 특히, 에러 처리에 신경을 쓰려고 했는데 맘처럼 잘 되지는 않았다. 그러나 프로그램중 드라이브가 준비되지 않았을 경우를 대비한 에러처리는 여러 프로그램에서 응용될 수 있다. 에러처리 함수의 구조는 다음과 같다.

```
int handler (int errval,int ax)
{
  char msg[25];
  int drive;

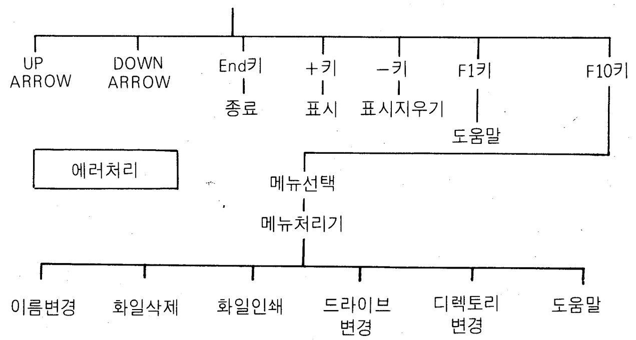
  if (ax < 0) {          /* device error */
    /* can only use dos functions 0 - 0x0C */
    bdosptr(DISPLAY_STRING, "device error$",0);
    hardretn(-1); /* return to calling program */
  }
  drive = ( ax & 0x00FF );
  gotoxy(5,24);
  sprintf(msg,"디스크 에러  : 드라이브 %c$ ",'A' + drive);
  bdosptr(DISPLAY_STRING, msg, 0);
  return(ABORT);         /* abort calling program */
```

다음은 메인 함수에서 이 프로그램을 설치하는 방법이다.

```
main()
{
  /* 프로그램의 맨 처음에 놓는다. */
  harderr(handler);
  ....
}
```

여기서 harderr라는 함수는 터보 C의 라이브러리 함수로 제공되고 있으며 도스 인터럽트 0x24가 요구될 때마다 프로

〔그림 1〕 프로그램의 구성



그램이 가지고 있는 에러 처리(여기서는 handler) 함수를 부르도록 해준다. 사용법은 void harderr(int(*fptr)())인데, 이때 fptr은 handler(int errval,int ax, int bp, int si)와 같은 인자들을 가지고 있어야 한다.

각 인자들에 대해서 설명하면 다음과 같다.

ax는 디스크에서 에러가 났는지 아니면 다른 디바이스에서 에러가 났는지를 지시한다. 만약, ax가 음수가 아니라면 디스크 에러이다. 만약, 디스크 에러가 났다면 이 ax의 값에 0x00FF를 And로 조작해 주면 에러가 난 드라이브의 번호를 알려준다. (1=A, 2=B, 등…)

bp, si는 둘다 모두 에러가 발생한 디바이스 드라이버의 헤더 포인터를 가지고 있다.

프로그램에 대해서 설명을 하면 먼저, 광역변수로 ffblks[..]를 선언했는데 이 변수에는 각 변수들의 이름, 날짜, 크기, 속성 등이 들어가며 그 형태는 표준 라이브러리와 같다.

그리고, ffatt[..]란 변수는 그 화일이 마크가 되었는지 안되었는지를 지시한다.

```
..
struct ffblk ffblks[300]
..
char ffatt[300]
            └──── 최대로 기억할 수 있는 화일의 갯수
                  숫자를 바꾸면 300개 이상의 화일목록도
                  읽어 들일 수 있다
```

화일 목록을 디스크로부터 읽는 함수는 makeffblks()란 함수로 이 함수에서는 findnext라는 함수를 사용하여 화일 목록을 읽는다.

```
void makeffblks()
{
  ...
  done=findfirst("*.*",&ffblk,FA_DIREC);
  /* 처음 화일목록을 읽는다. FA_DIREC은 읽어들일 화일목록들의 */
  /* 속성을 의미하는데, 여기서는 서브 디렉토리까지 읽어들인다. */
  ....
  while(!done) {
  ....
  done=findnext(&ffblk);
  /* 다음번 화일목록을 읽어들인다. 만약, 더 읽을 화일이 없으면   */
  /* 0 값을 리턴해 준다. */
  ...
  }
}
```

화일을 화면으로 출력해 주는 함수는 view()이며 계속 키보드를 조사해서 만약 ESC키가 눌려졌을 경우 출력을 중지하고 다른 키가 눌려졌을 경우 잠시 멈추게 해준다. 이 방법을 응용하면 유저의 키보드 체크를 컴퓨터가 다른 일을 하는 동안에도 할 수 있게 해준다. (굳이 인터럽트 벡터를 바꾸지 않아도)

```
int view(char *filename)
{
  ....
  while(!feof(fp)) {
   ....
   chkbd=bioskey(1);   /* 키보드가 눌렸는지 검사한다.. */
   if (chkbd!=0) {     /* 키보드가 눌렸을 경우 다음 루틴을 처리한다. */
     chkbd=bioskey(0); /* 키보드 버퍼로 부터 한 글자를 가져온다. */
     ....              /* 키보드가 눌렸을때 처리해야 하는 루틴이 있다. */
   }
   .... /* 컴퓨터가 처리해야하는 루틴이 있다. */
}
```

약간의 수정이 필요한 곳이 있는 함수들은 renamefile(), changedir() 등인데 이 함수들에서 renamefile()의 예를 들어서 어떤 곳을 바꾸어야 하는지 살펴보자.

```
int renamefile(int num)
{
 ...
 scanf("%s",buf); /* 바로 이 함수가 바뀌어야 한다.*/
 ...
 ...
}
```

scanf를 사용해도 별무리 없이 프로그램이 실행되지만 보다 잘 작성한 프로그램이 되려면 독자적인 키보드 입력 함수를 가지고 있어야 한다. 즉, scanf 함수에서는 ESC키나 break키가 작동하기 때문에, 적당한 크기 이상으로 키 입력을 받을 경우 화면이 깨지는 경우가 발생하므로 이러한 일들을 처리해 줄 수 있는 독자적인 키 입력함수가 필요하다.

이러한 함수를 작성하는 방법은 그다지 어렵지 않으므로 여기서는 소개하지 않겠다. 약간의 힌트를 드린다면 getch() 함수는 break 키도 입력받지 않고, 입력된 키를 화면으로 에코해 주지도 않기 때문에 이 함수를 이용하여 작성하면 된다.

알고리즘

사용된 알고리즘은 지난 호의 윈도우 함수처럼 동적인 메모리 관리를 사용하지 않았고 정적인 메모리 관리를 사용했다. 이렇게 정적인 메모리 관리를 선택한 이유는 프로그램에서의 메모리를 효율적으로 관리하기 위해서이다.

동적인 메모리 관리는 메모리를 프로그램이 실행되기 전

에 지정하므로 하나의 메모리 단위가 상당히 클 경우에 큰 이점을 가진다. 즉, 한 사람의 신상명세를 보관하는데 필요한 공간이 약 2K바이트 정도라고 생각했을 경우 정적으로 메모리 관리를 해서 200명 정도를 보관하려고 하면 400K바이트가 필요하다.

IBM PC는 640K 정도의 메모리를 가지고 있지만 사이드킥과 같은 램상주 프로그램 몇개만 실행시키면 유저가 사용 가능한 메모리는 400K 이하가 되어버린다. 이런 경우 한꺼번에 필요한 메모리 전체를 할당받는 정적인 메모리 관리를 사용한 프로그램은 out of memory라는 메시지를 출력하고 실행되지 않는다.

그러나 필요할 때마다 도스로부터 메모리를 할당받는 동적인 메모리 관리는 그러한 걱정이 없지만 동적인 메모리 관리가 가지는 약점은 정적인 메모리 관리에서 필요없는 메모리를 동적인 메모리 관리는 필요로 한다는 것이다.

이러한 필요없는 메모리의 양은 어떠한 자료구조이냐에 따라 달라지고 또, 어떠한 메모리 모델을 사용했느냐에 따라 달라지는데 일반적으로 Linked List 구조일 경우 약 5~8바이트 정도가 필요하다. 물론 데이터 하나가 2000바이트 정도가 되면 이 정도야 새발에 피지만 데이터가 20바이트라면 문제가 생긴다. 자료의 25% 이상이 되는 양의 메모리를 낭비하게 되는 것이다. 이러한 경우는 정적인 메모리 관리를 사용하는 것이 훨씬 유리하다.

정적인 메모리와 동적인 메모리의 차이점은 더 많이 있지만 그러한 내용은 다음 기회에 다루기로 하고 이곳에서는 단지 위와 같은 이유로 정적인 메모리 관리(Array 사용)를 택했다는 사실만을 염두하기 바란다.

프로그램의 사용법

본 프로그램을 이용하게 될 때 노턴 코맨더를 써보신 유저분들은 아마도 금방 익숙해질 것이다. 그러나, 노턴 코맨더처럼 기능이 완전하지는 못하다.

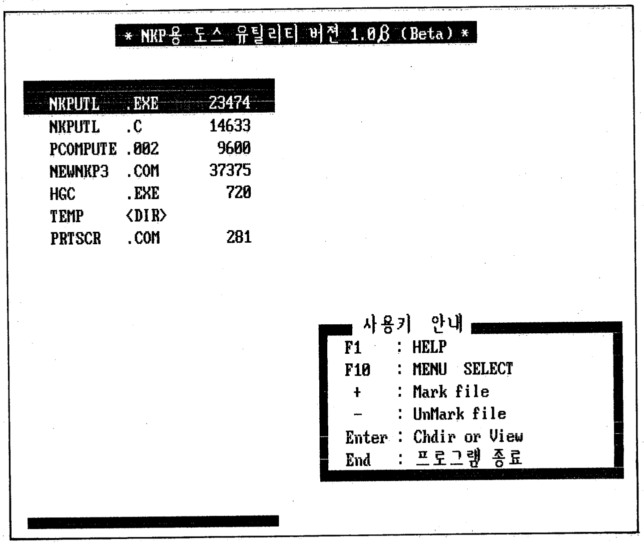
우선 사용 드라이브상에 HGC FULL과 NKP를 실행시킨 후 NKPUTL를 입력하여 프로그램을 실행시키면 로고 화면이 나온다.

그리고 아무키나 누르면 **화면** 1에서 보는 바와 같은 실행화면이 나타난다. 이때 위로 가는 화살표나 아래로 가는 화살표를 눌러서 화일을 선택할 수가 있다.

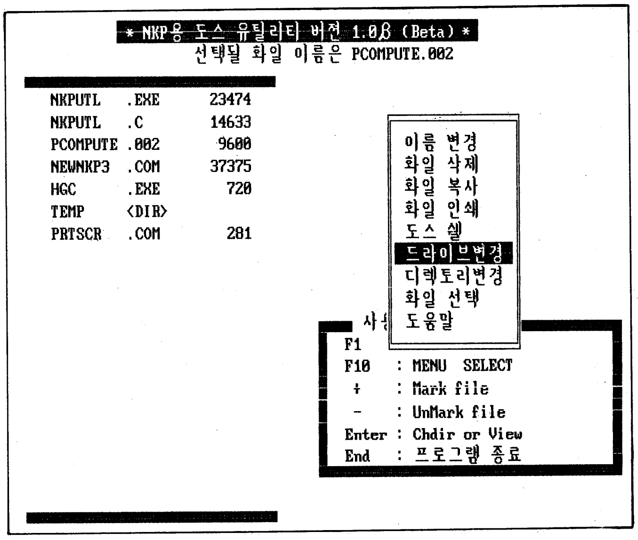
화일을 표시(Mark)하고 싶으면 +키를 누르면 된다. 다시 표시를 지우고 싶으면 −키를 누른다. 이러한 표시 기능은 나중에 메인 메뉴에서 복수의 화일을 선택하여 지울때 사용이 될뿐 아니라, 차후에 기능을 확장할 때도 요긴하게 쓰일 수 있다.

특징적인 기능을 보면 화일에 반전 막대가 놓여 있을 때 리턴키를 누르면 그 화일을 출력해 준다.

〔화면 1〕 실행화면



〔화면 2〕 메뉴 화면



만약, 화일이 아니라 서브 디렉토리일 경우에는 그 디렉토리로 이동한다. 그리고 F10키를 누르면 메인메뉴(?)가 나오게 되어 있다. 화면구성은 팝업 메뉴방식으로 되어 있으며 각 메뉴들의 기능을 살펴보면 다음과 같다.

이름변경 : 이 기능은 화일의 이름을 바꾸어 준다. 일단 이 기능을 선택하면 대화 윈도우가 생기면서 현재 선택된 화일의 이름을 보여주고 바꿀 화일의 이름을 입력받는다.

화일삭제 : 화일을 삭제하는 기능

화일인쇄 : 화일을 프린터로 인쇄해 준다. View기능과 거의 비슷하다.

드라이브 변경 : 드라이브를 변경시켜 준다. 현재, 사용중인 드

〔화면 3〕 디렉토리 변경 선택후의 화면

```
         * NKP용 도스 유틸리티 버전 1.0β (Beta) *
                선택될 화일 이름은 PCOMPUTE.002

  NKPUTL   .EXE     23474
  NKPUTL   .C       14633
  PCOMPUTE .002      9600
  NEWNKP3  .COM     37375
  HGC      .EXE       720
  TEMP     <DIR>
  PRTSCR   .COM       281

  현재 디렉토리는  A:\
  바꿀 디렉토리는  > _
                             ㅎ키  안내
                                : HELP
                                : MENU  SELECT
                          +     : Mark file
                          -     : UnMark file
                          Enter : Chdir or View
                          End   : 프로그램 종료
```

라이브의 이름을 출력해주면서 바꿀 드라이브 이름을 입력받는다.

디렉토리 변경 : 디렉토리를 변경시켜 준다. 드라이브 변경과 마찬가지로 현재 사용중인 드라이브의 이름을 출력해주고, 바꿀 드라이브 이름을 입력받는다.(참고로 화면 2와 3에 이의 실행 예를 보인다)

혹, 위의 내용을 잊어버리더라도 걱정하지 않아도 된다. 어디서나 도움말 화면이 제공되기 때문이다.

이상으로 프로그램에 대한 설명을 마치기로 하고 혹, 의문나는 점 등이 있는 독자 여러분은 필자가 운영하고 있는 온라인 BBS Ⅱ로 연락하기 바란다. 전화번호는 453-4491이고 운영시간은 저녁 10시부터 아침 7시까지이므로 본 프로그램에 관한 것이 아니라도 컴퓨터에 관한 여러 의문사항에 관하여 문의하기 바란다. PC

〔리스트 1〕 NKPUTL.C

```
#include <stdio.h>
#include <conio.h>
#include <dir.h>
#include <dos.h>
#include <alloc.h>
#include <bios.h>
#include <io.h>

#include <d:\tc\work\winlib.c>
/*          _________ -> directory of winlib.c */

#define DISPLAY_STRING 0x09
#define IGNORE 0
#define RETRY  1
#define ABORT  2

  struct ffblk ffblks[300];
  int totalnum;
  char ffatt[300];

/* header block definition part */
/* ------------------ hardware error handler ---------------- */
int handler (int errval,int ax)
{
  char msg[25];
  int drive;

  if (ax < 0) {          /* device error */
    /* can only use dos functions 0 - 0x0C */
    bdosptr(DISPLAY_STRING, "device error$",0);
    hardretn(-1); /* return to calling program */
  }
  drive = ( ax & 0x00FF );
  gotoxy(5,24);
  sprintf(msg,"디스크 에러  : 드라이브 %c$",'A' + drive);
  bdosptr(DISPLAY_STRING, msg, 0);
  return(ABORT);         /* abort calling program */
}
/* ---------------- Display Error message -------------------- */
void errors(int err)
{
  openwindow(5,30,17,50,5,2);
  gotoxywin(1,1);
  printf("      에러가 발생했습니다 !!");
  gotoxywin(1,2);
  switch(err) {
   case 1 : printf(" 화일이름을 바꿀 수가 없습니다."); break;
   case 2 : printf(" 이 화일을 삭제할 수가 없습니다."); break;
   case 3 : printf(" Copy file is not installed. Sorry !"); break;
   case 4 : printf(" 화일을 출력할 수가 없습니다."); break;
   case 5 : printf(" 이 기능은 아직 만들어지지 않았습니다");     break;
   case 6 : printf(" 드라이브 전환이 안됩니다  !");    break;
   case 7 : printf(" 디렉토리 전환이 안됩니다 !"); break;
  }
  gotoxywin(1,3); printf(" << 임의의 키를 누르시면 계속됩니다 >>");
  getch();
  closewindow(5,30,17,50,5,2);
}

/* -------------- Help Funtion ------------------------------- */
void help(int num)
{
  switch(num) {
   case 1 :
```

```
    openwindow(1,25,5,42,14,2);
    gotoxywin(10,1);printf(" * 도움말 화면 1 * ");
    gotoxywin(1,3); printf(" End키 : 프로그램의 종료 ");
    gotoxywin(1,4); printf("  + 키 : 화일을 선택합니다.");
    gotoxywin(1,5); printf("  - 키 : 선택된 화일을 취소합니다.");
    gotoxywin(1,6); printf(" F1 키 : 도움말 화면");
    gotoxywin(1,7); printf(" F10키 : 주요 기능들을 선택하게 해줍니다");
    gotoxywin(1,8); printf(" Enter 키 : Chdir - 디렉토리의 전환");
    gotoxywin(1,9); printf("         : View  - 문자화일의 출력");
    gotoxywin(1,11);printf("                      프로그램 by 구 준회 ");
    gotoxywin(3,12);printf("<<임의의 키를 누르면 계속됩니다.>>");
    getch();
    closewindow(1,25,5,40,10);
    break;
   case 2 :
    openwindow(1,10,5,42,14,2);
    gotoxywin(10,1);printf(" * 도움말 화면 2 * ");
    gotoxywin(1,3); printf(" 이름 변경 : 화일의 이름을 바꾸어줍니다");
    gotoxywin(1,4); printf(" 화일 삭제 : 화일을 삭제합니다          ");
    gotoxywin(1,5); printf(" 화일 복사 : 화일을 복사해줍니다        ");
    gotoxywin(1,6); printf(" 화일 인쇄 : 문자 화일을 프린터로 인쇄 ");
    gotoxywin(1,7); printf(" 도스 쉘    : 도스로 빠져나갑니다         ");
    gotoxywin(1,8); printf(" 디라이브 변경 : 사용 드라이브 전환");
    gotoxywin(1,9); printf(" 디렉토리 변경 : 사용 디렉토리 전환");
    gotoxywin(1,10);printf(" 화일 선택 : 화일을 선택하도록 합니다  ");
    gotoxywin(1,11);printf(" 도움말      : 현재의 화면을 보여줍니다  ");
    gotoxywin(1,12);printf(" << 임의의 키를 누르시면 계속 됩니다 >>");
    getch();
    closewindow(1,10,5,40,10);
    break;
   default: break;
  }
}
/* --------------- Make file direcotry array ------------------ */
void makeffblks()
{
  struct ffblk ffblk; /* file block structure */
  int done,num=0,i;

  done= findfirst("*.*",&ffblk,FA_DIREC);
  /* searches disk for directory*/
  ffblks[num]=ffblk;
  while(!done) {
    num++;
    done=findnext(&ffblk);
    ffblks[num]=ffblk;
    ffatt[num]=0;
  }

  totalnum=num;
}
/* ---------------- Display file directory ------------------- */
/* startfn is start file number in file block array */
int displayf(int startfn)
{
  int i,temp=0;
  char temp1[3],temp2[66],filename[9],filext[5];

  if (startfn<0) { startfn=0; }
  if (totalnum>19) {
    if (startfn+19>totalnum) { startfn=totalnum-19; }
  }
  /* check start file number is correct */
  for(i=startfn;i<totalnum;i++) {
    gotoxy(3,4+temp);
    if (ffatt[i]==1) { printf("•"); } else { printf("  "); }
    gotoxy(5,4+temp);
    if (ffblks[i].ff_attrib!=FA_DIREC) {  /* display file */
      fnsplit(ffblks[i].ff_name,temp1,temp2,filename,filext);
      printf("%-9s%-4s %10lu",filename,filext,ffblks[i].ff_fsize);
    }
    else {
      printf("%-9s<DIR>                ",ffblks[i].ff_name);
    }
   temp++;
   if (temp>=19) break;
  }
  if (totalnum<19) { for(;i<19;i++) {
    gotoxy(3,4+i); printf("                          "); }
  }
  return(startfn);
}

/* ---------------- Veiw or Change Directory ----------------- */
int view(char *filename)
{
  FILE *fp;
  char ch,chkbd;

  fp=fopen(filename,"r");
  while(!feof(fp)) {
    ch=fgetc(fp);
    chkbd=bioskey(1);
    if (chkbd!=0) {      /* check key buffer */
      chkbd=bioskey(0); /* get character from key buffer */
      if (chkbd==27) { return(0); }
      else {
        printf("\n임의의 키를 누르시면 계속됩니다....(ESC: 종료)\n");
        getch();
      }
    }
    printf("%c",ch);
  }
   printf("\n임의의 키를 누르시면 계속됩니다....\n");
   getch();
   fclose(fp);
```

```
}
/* Same as view but print */
int viewp(char *filename)
{
  FILE *fp;
  char ch,chkbd;

  fp=fopen(filename,"r");
  while(!feof(fp)) {
    ch=fgetc(fp);
    chkbd=bioskey(1);
    if (chkbd!=0) {     /* check key buffer */
      chkbd=bioskey(0); /* get character from key buffer */
      if (chkbd==27) { return(0); }
      else {
        gotoxywin(1,2);
        printf("임의의 키를 누르시면 계속됩니다.");
        gotoxywin(1,3);printf(" ESC for stop ");
        getch();
      }
    }
    fprintf(stdprn,"%c",ch);
  }
  fclose(fp);
}

int viewchdir(int num)
{
  if (ffblks[num].ff_attrib==FA_DIREC) {
    if(chdir(ffblks[num].ff_name)) { errors(6); }
    makeffblks();
    return(1); /* Change directory processed */
  }
  else {
    clrscr();
    view(ffblks[num].ff_name);
    return(0);
  }
}

/* ----------------- Menu Selector -------------------- */
/* rename file       */
/* return 1 if renamed  else return 0 */
int renamefile(int num)
{
  char buf[13];
  int result;

  openwindow(2,5,12,40,5,1);
  gotoxywin(1,1); printf("  현재 화일 이름은  %s",ffblks[num].ff name);
  gotoxywin(1,2); printf("  바꿀 이름은 > ");
  scanf("%s",buf);  /* But this function will be changed */
  result=rename(ffblks[num].ff_name,buf);
  if (result==-1) { errors(2); result=0; /* error message */ }
  else result=1;
  closewindow(2,5,12,40,5,1);
  return(result);
}

/* delete file(s) */
/* return 1 if deleted else return 0 */
int deletefile(int num)
{
  int i,result, selected=0;

  openwindow(2,5,12,40,5,1);
  gotoxywin(1,1); printf(" 화일을 삭제합니다. 확실합니까(y/n) ?");
  if (getch()=='y') {
    printf("y");
    for(i=0;i<totalnum+1;i++) {
      if (ffatt[i]==1) {
        selected=1;
        gotoxywin(1,2); printf("삭제하는 화일 이름은 %s",
                                              ffblks[i].ff name);
        result=unlink(ffblks[i].ff_name);
        if (result==-1) { errors(1); result=0; /* error message */ }
        else result=1;
      }
    }
    if(selected!=1) {
      gotoxywin(1,2); printf("삭제하는 화일 이름은 %s",
                                              ffblks[num].ff name);
      result=unlink(ffblks[num].ff_name);
      if (result==-1) { errors(3); result=0; /* error message */ }
      else result=1;
    }
  }
  closewindow(2,5,12,40,5,1);
  return(result);
}

/* print file */
void printfile(int num)
{
  char ch;
  openwindow(2,5,12,40,5,1);
  gotoxywin(1,1); printf("인쇄할 화일 이름은  %s",ffblks[num].ff name);
  gotoxywin(1,2); printf("프린터가 준비 됐읍니까 ? (y/n) ");
  ch=getch();
  if (ch=='y')  viewp(ffblks[num].ff_name);
  closewindow(2,5,12,40,5,1);
}

/* Drive Change  */
/* return 1 if changed  else return 0 */
int changedrive()
```

```
{
  int buf,result,target;

  openwindow(2,5,12,40,5,1);
  gotoxywin(1,1); printf("사용중인 드라이브 이름은 %c 입니다",
                          getdisk()+65);
  gotoxywin(1,2); printf("바꿀 드라이브 이름은 > ");
  buf=getch();
  buf=toupper(buf);
  printf("%c",buf);
  target=buf-65;  /* make drive number */
  result=setdisk(target);
  if (result==-1) { errors(5); result=0; /* error message */ }
  else result=1;
  closewindow(2,5,12,40,5,1);
  return(result);
}
/* change directroy  */
/* return 1 if changed else return 0 */
int changedir()
{
  char buf[126];
  int result;

  openwindow(2,5,12,40,5,1);
  getcwd(buf,126);
  gotoxywin(1,1); printf("현재 디렉토리는  %s",buf);
  gotoxywin(1,2); printf("바꿀 디렉토리는  > ");
  scanf("%s",buf);  /* But this function will be changed */
  result=chdir(buf);
  if (result==-1) { errors(7); result=0; /* error message */ }
  else result=1;
  closewindow(2,5,12,40,5,1);
  return(result);
}

/* menu selector (MAIN) */
int menuselector()
{
  int result;

  openwindow(1,45,5,16,11,2);
  gotoxywin(1,1); printf(" 이름 변경");
  gotoxywin(1,2); printf(" 화일 삭제");
  gotoxywin(1,3); printf(" 화일 복사");
  gotoxywin(1,4); printf(" 화일 인쇄");
  gotoxywin(1,5); printf(" 도스 쉘 ");
  gotoxywin(1,6); printf(" 드라이브변경");
  gotoxywin(1,7); printf(" 디렉토리변경");
  gotoxywin(1,8); printf(" 화일 선택");
  gotoxywin(1,9); printf(" 도움말 ");
  result=selectmenu();
  closewindow(1,45,5,14,12);
  return(result);
}
/* --------------- Menu Processor ------------------------ */
/* if file block  revoer is need, return 1                  */
/* if file block  revoer is not need, return 0              */
int processmenu(int menu, int num)
{
   int result;

   switch(menu) {
     case 1 : result=renamefile(num);return(result); /* rename */
     case 2 : result=deletefile(num);return(result); /* delete */
     case 3 : /* copy */ break;
     case 4 : printfile(num); return(0);/* print file */
     case 5 : /* OS shell */   break;
     case 6 : result=changedrive();  return(result);
                                              /* Change Drive */

     case 7 : result=changedir(); return(result);
              /* Change Driectory */
     case 8 : return (0);
     case 9 : help(2); break;
   }
}

/* --------------- Make Log --------------------- */
void makelog()
{
  int i;

  clrscr();
  gotoxy(13,1); printf(" * NKP용 도스 유틸리티 버젼 1.0β (Beta) * ");
  changeatt(13,1,43);
  for(i=1;i<16;i++) {
    gotoxy(i*2,3);  printf("■");
    gotoxy(i*2,23); printf("■");
  }
  gotoxy(37,14); printf("■■ 사용키 안내 ■■■■■■■■■");
  gotoxy(37,15); printf("▌ F1    : HELP          ▌");
  gotoxy(37,16); printf("▌ F10   : MENU SELECT   ▌");
  gotoxy(37,17); printf("▌  +    : Mark file     ▌");
  gotoxy(37,18); printf("▌  -    : UnMark file   ▌");
  gotoxy(37,19); printf("▌ Enter : Chdir or View ▌");
  gotoxy(37,20); printf("▌ End   : 프로그램 종료   ▌");
  gotoxy(37,21); printf("■■■■■■■■■■■■■■■■■■■■");
}

void initlog()
{
  int i;

  clrscr();
  gotoxy(1,5);
```

```
printf("     ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■\n");
printf("     █   NKP 용 도스 유틸리티 프로그램 버전 1.0β (베타)   █\n");
printf("     █■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■  █\n");
printf("     █ 프로그램 by 구 준회                                █\n");
printf("     █            고려 대학교 컴퓨터 클럽                   █\n");
printf("     █            eXtended System Engineering Association █\n");
printf("     █                                                    █\n");
printf("     █ 연락처 : 온라인 BBS I   603-1835  (21:00-06:00)     █\n");
printf("     █          온라인 BBS II  453-4491  (22:00-07:00)     █\n");
printf("     █                                                    █\n");
printf("     ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■\n");
getch();
}
main()
{
  int ch,filenum=0,cursor_y=0,result;

  harderr(handler);  /* establishes a hardware error handler */

  initlog();
  makelog();
  makeffblks();
  filenum=displayf(0);   /* initialize program */
  changeatt(2,4,30);
  for(;;) {
    ch=getarrow();
    changeatt(2,4+cursor_y,30);
    switch(ch) {
      case  4 :
      case  8 :          /* Key pressed Left or Up */
        cursor_y--;
        if (cursor_y<0) {
          cursor_y=18;
          if (cursor_y>totalnum-1) cursor_y=totalnum-1;
          /* check end of directory entery */
          filenum=filenum-18;
          if (filenum<0) filenum=0;
          filenum=displayf(filenum);
        }
        break;
      case  6 :
      case  2 :          /* Key pressed Right or Down */
        cursor_y++;
        if (cursor_y>totalnum-1) cursor_y=totalnum-1;
        /* check end of directory entry */
        if (cursor_y>18) {
          cursor_y=0;
          filenum=filenum+18;
          filenum=displayf(filenum);
        }
        break;
      case 43 :          /* '+' key pressed. Mark file */
        if (ffblks[filenum+cursor_y].ff_attrib!=FA_DIREC) {
          gotoxy(3,4+cursor_y); printf("•");
          ffatt[filenum+cursor_y]=1;
        }
        break;
      case 0x2d :        /* '-' key pressed. Mark file */
        gotoxy(3,4+cursor_y); printf(" ");
        ffatt[filenum+cursor_y]=0;
        break;
      case 13 :          /* C/R key pressed view or Change directory */
        result=viewchdir(filenum+cursor_y);
        /* result 1 is chdir, result 0 is view */
        if (result==1) {filenum=0;cursor_y=0;}
        else { makelog(); }
        displayf(filenum);
      break;
      case 59 : /* F1 key pressed ( Help ) */
        help(1);
        break;
      case 68 : /* F10 key pressed ( Menu Selector ) */
        result=menuselector();
        result=processmenu(result,filenum+cursor_y);
        if (result==1) {
          makeffblks();
          filenum=0; cursor_y=0;
          displayf(0);
        }
          break;
      case  1 : exit(0); break;
    }
    changeatt(2,4+cursor_y,30);
    gotoxy(22,2);
    printf("선택될 화일 이름은 %s ",ffblks[filenum+cursor_y].ff_name);
  }
}
```

인터럽트 핸들러

방재근 / 한양대 HY-CoRA

프로그램실행시 좀더 유용한 동작을 위하여 인터럽트를 이용할 수 있다. 여기서 소개하는 인터럽트 핸들러도 몇가지 함수를 이용한 프로그램으로 다른 프로그램실행시 시간의 출력이나 키상태 표시등을 할 수 있는 인터럽트 응용프로그램이다.

필자는 지난 겨울 방학을 이용하여 아르바이트로 프로그램개발에 참여하였는데, 그때 담당한 부분은 화면 디스플레이와 프린터 출력의 I/O 부분이었다. 이 부분에 대한 프로그램을 짜면서 제기된 문제는 프로그램 실행중 데이터 입력상태로 되어있는 경우에도 현재 시간을 화면 우측상단에 계속해서 표시하고 입력된 키의 상태를 우측 하단에 표시하는 것이었다.

그러나 입력루틴이나 내부처리, 프린터 출력 등의 프로그램이 실행중일 때에는 시간 표시의 정지와 Caps, Num, Scroll 키 등의 입력상태 체크를 일시적으로 멈추어 화면 출력이 정지된 상태가 생겨 그리 깔끔하지 못하였다. 물론 이것에 대한 테크닉은 갖고 있었지만 그 당시 전체 프로그램으로 볼 때 비효율적이라는 판단과 화면을 표시하는데 있어서의 조잡성 등의 이유로 결국 전체프로그램에서 제외시켰었다.

그런데 볼랜드사의 터보시리즈의 통합환경을 쓰면서 항상 느낀 것은 언제, 어느 상태에서도 현재의 키 상태(Caps, Num, Scroll)를 화면 우측 하단에 나타낼 수 있다는 것이다. 그래서 이러한 현상을 어떻게 처리했을까하는 궁금증과 호기심으로 작성해 본 것이 '인터럽트 핸들러'이다. 이 '인터럽트 핸들러'는 파스칼언어로 작성되었으며 이 프로그램이 어떻게 사용되는지 그 원리와 기법을 용어설명과 함께 소개하기로 하겠다.

인터럽트

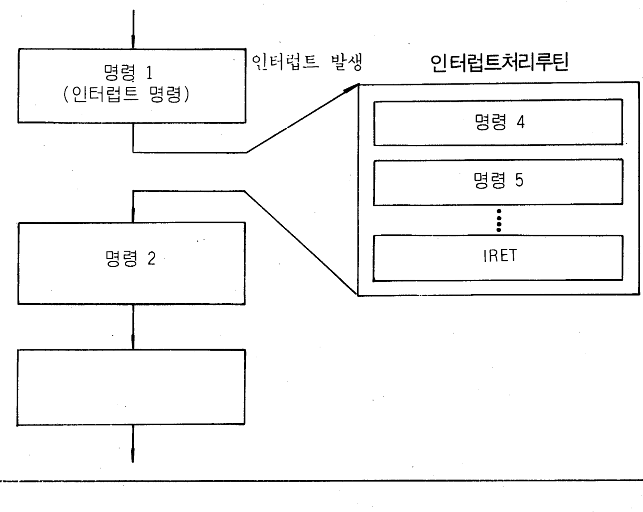
먼저 인터럽트가 무엇이며, 어느때 발생하고 어떠한 종류가 있는지 살펴보기로 하자. 인터럽트란 내부 혹은 외부의 어떤 긴급을 요하는 원인이 발생하여 현재 수행하고 있는 메인 프로그램을 일시 중단하고 해당 루틴인 인터럽트 처리루틴을 수행하여 다음에 재개할 수 있게끔 한다음, 다른 프로그램을 실행할 수 있도록 만들어 주는 것으로 이 프로그램이 끝나면 원래의 메인 프로그램으로 돌아갈 수 있도록 해준다(그림 1 참조).

이 인터럽트는 0부터 255까지 지정된 인터럽트가 있으며, CPU의 내부에서 발생하는 내부 인터럽트와 CPU의 외부에서 발생하는 외부 인터럽트로 나눌 수 있다. 내부 인터럽트에는 타이머 인터럽트나 메모리 패리티 에러, 전원의 비정상등이 원인이 되는 하드웨어의 에러 인터럽트가 있으며, 0으로 나눗셈 계산을 하였을 때와 일종의 서브루틴 호출로 인해 발생하는 소프트웨어 인터럽트 등이 있다. 그리고 외부 인터럽트에는 주변 장치의 고장과 같은 입출력 동작에 기인하여 발생하는 인터럽트와 그 동작 완료나 이상을 통지하는 입출력 인터럽트, 그리고 프로세서 인터럽트 등이 있으며, 이 밖에 제어 시스템에서 피제어계로부터 발생하는 인터럽트 등이 있다.

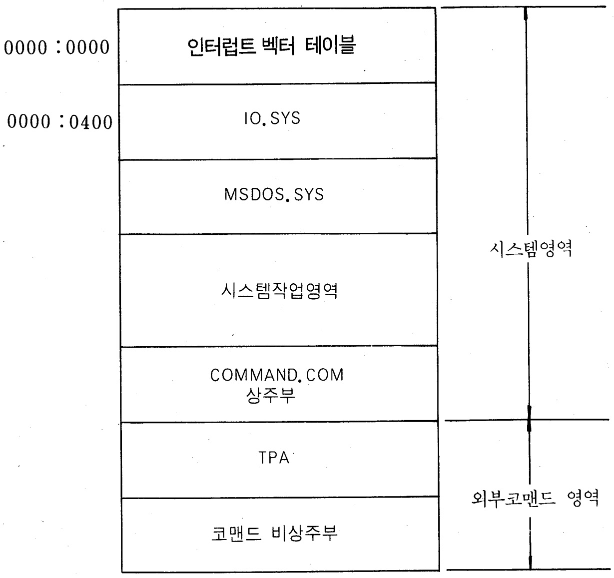
인터럽트 벡터

소프트웨어 인터럽트 혹은 하드웨어 인터럽트가 발생하였

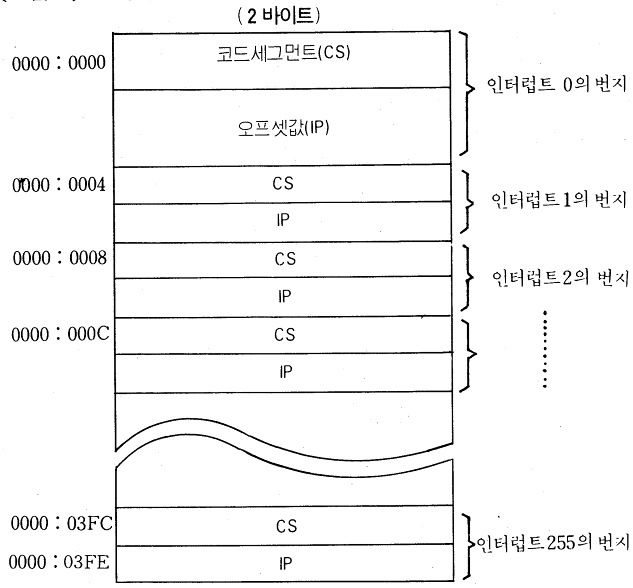
〔그림 1〕 인터럽터 발생시 실행순서



〔그림 2〕 메모리 구성



〔그림 3〕 인터럽트 벡터 테이블



을 때에 CPU는 현재 실행중인 프로그램을 중단하고 인터럽트 처리루틴으로 실행을 옮긴다고 위에서 언급하였다. 그때 실행을 옮기는 인터럽트 처리루틴의 번지가 씌어져 있는 곳을 인터럽트 테이블 혹은 인터럽트 벡터라고 한다.

인터럽트는 전부 256종류가 있다고 하였으며, 인터럽트 처리루틴으로 옮겨 가려면 그 루틴에 있는 세그먼트 번지와 오프셋 번지를 통털어 4바이트가 필요하여 결과적으로 인터럽트 벡터는 4×256=1024 바이트가 소요된다. 인터럽트는 각각의 해당 번호로 실행이 되는데 그 메모리 구성은 **그림 2**에 나타내었다. 여기서 번지 0부터 3FFH(16진수)가 바로 인터럽트 벡터 테이블이다. 여기에 0부터 255까지의 인터럽트 각각의 처리루틴 번지가 순서대로 있어 인터럽트가 발생하면 이곳으로 점프하였다가 해당 인터럽트의 번지로 또다시 점프하여 루틴을 실행하게 되는 것이다.

그 구조는 위에서 설명했듯이 각 인터럽트마다 4바이트를 차지하는데 처음 2바이트는 처리루틴의 코드 세그먼트, 다음 2바이트는 오프셋값을 나타낸다. **그림 3**은 인터럽트 벡터 테이블을 알기 쉽게 표시한 것이고, **화면 1**은 디버거로 표시한 것을 나타낸 것이다. 루틴번지의 계산은 코드 세그먼트의 실제 물리적인 번지인 세그먼트 베이스에 오프셋값을 더해서 구한다. 여기서 실제 물리적인 번지란 세그먼트 베이스에 0의 자리를 하나 추가시킨 번지를 말한다. 예를 들어 코드 세그먼트가 2000H, 오프셋이 3456H일 경우 세그먼트 베이스는 20000H가 되어 루틴 번지는 23456H가 되는 것이다.

타이머 인터럽트

시스템은 1초에 18.2번씩 정지하여 인터럽트 8H를 실행하는데 이것의 역할은 시스템의 내부시간을 갱신시키고, 날짜가 바뀌었는가를 검사하며, 또 다른 인터럽트 1CH를 호출하여 타이머를 제어한다. 타이머칩 8253은 매 0.0549254초마다 메모리 40:6C와 40:6E에 있는 값을 증가하여 시간을 갱신시킨다. 다시 말해서 시스템이 1초에 18.2번 정지하는 이유가 되는 것이다. 6C의 값이 증가하여 6D의 값을 증가시

〔화면 1〕 인터럽트 벡터 테이블 덤프

```
C:\DOS>DEBUG
-D0:0
0000:0000  E8 56 61 02 56 07 70 00-16 00 AD 0B 56 07 70 00   .Va.V.p.....V.p.
0000:0010  56 07 70 00 54 FF 00 F0-23 FF 00 F0 23 FF 00 F0   V.p.T...#...#...
0000:0020  E1 07 73 0D 50 09 73 0D-23 FF 00 F0 23 FF 00 F0   ..s.P.s.#...#...
0000:0030  23 FF 00 F0 B1 03 AD 0B-39 04 AD 0B 56 07 70 00   #.......9...V.p.
0000:0040  03 00 73 0D 4D F8 00 F0-41 F8 00 F0 7A 0F 70 00   ..s.M...A...z.p.
0000:0050  39 E7 00 F0 59 F8 00 F0-2E E8 00 F0 D2 EF 00 F0   9...Y...........
0000:0060  00 00 00 F6 E0 18 70 00-6E FE 00 F0 50 07 70 00   ......p.n...P.p.
0000:0070  53 FF 00 F0 A4 F0 00 F0-22 05 00 00 01 16 73 0D   S.......".....s.

-D0:380
0000:0380  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0000:0390  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0000:03A0  00 00 00 00 00 00 00 00-00 00 00 00 EB FE 00 F0   ................
0000:03B0  97 F2 01 08 F2 03 00 08-06 00 02 08 F4 03 02 FF   ................
0000:03C0  26 ED 39 EF 48 08 88 EC-00 C8 00 C8 00 C8 60 09   &.9.H.........₩.
0000:03D0  AC 03 00 C8 DB 02 00 C8-60 09 99 BA 00 C8 6C 03   ........₩.....l.
0000:03E0  00 00 00 00 00 00 3C E0-04 00 00 00 80 00 01 00   ......<.........
0000:03F0  00 7C B6 02 00 C8 86 F2-01 00 F6 E4 00 F0 46 F2   .¦............F.
```

키고, 다시 6E의 값을 증가시켜 결국 6E에 들어있는 16진 수값이 몇시인가를 나타내게 되는 것이다. DEBUG 프로그램으로 **화면 2**를 실행하면 확인할 수 있을 것이다.

인터럽트 핸들러의 원리

인터럽트가 발생하면 위에서 언급했듯이 인터럽트 벡터 테이블로 점프하여 해당번지에 있는 코드세그먼트와 오프셋 값을 다시 메모리번지로 하여 해당 루틴을 실행하고 처음 인터럽트가 발생된 루틴에서 그 다음 부분을 계속해서 실행하는데, 여기서 인터럽트 벡터 테이블내의 해당 번지에 있는 코드세그먼트와 오프셋값 대신 프로그래머가 새로 작성한 루틴 번지로 변경하면 해당 인터럽트가 발생할 때 마다 새로 기록된 번지의 루틴이 실행되는 것이다.

이해를.돕기위해 **그림 4**에 이러한 과정을 도식으로 표시하였다. 그림 4(a)의 그림은 원래 프로그램 실행중 인터럽트가 생겼을 때의 순서이고, (b)의 그림은 새로운 프로그램을 삽입함으로써 루틴 번지가 변경되어 실행하는 순서를 나타낸 것이다. **화면 3**은 램상주 프로그램 혹은 새로운 인터럽트 루틴이 실행되기 전과 후에 INT 1CH에 있는 번지가 변경되어 있음을 보여준다.

인터럽트 핸들러의 기법

PC 언어중 많이 사용하고 있는 볼랜드사의 터보파스칼은 자체에 인터럽트 핸들러를 라이브러리로 지원하고 있다. 다음은 라이브러리로 지원하는 프로시듀어를 나타낸 것이다.

GetIntVec(IntNo : byte ; var Vector : pointer)

기능 : 인터럽트 실행루틴 번지(코드세그먼트 : 오프셋값)를 얻음

IntNo : 인터럽트 번호(0<=IntNo<=255)
Vector : 실행 루틴 번지

SetIntVec(IntNo : byte ; Vector : pointer)

기능 : 인터럽트 실행루틴 번지를 인터럽트 벡터 테이블에 기록

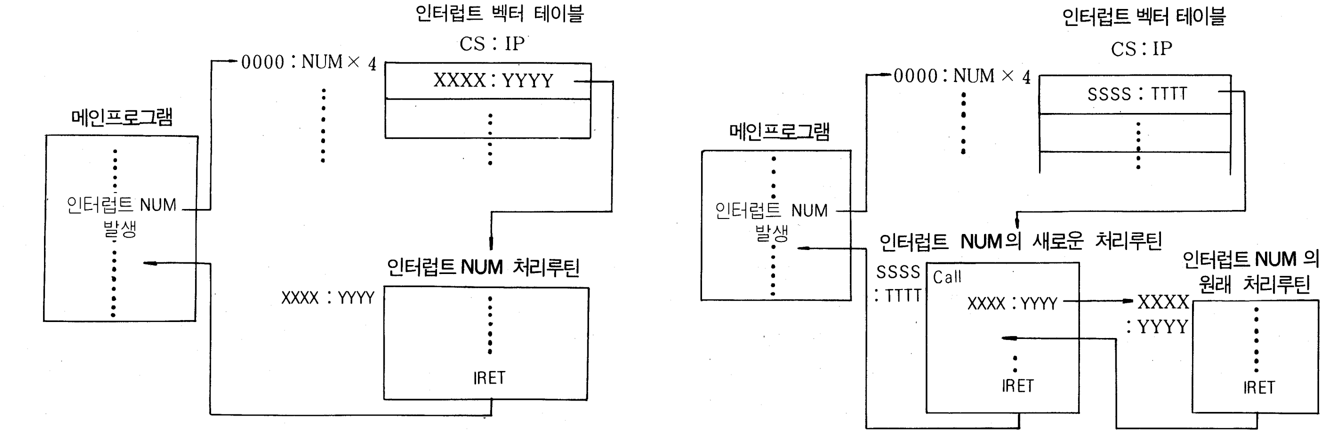
IntNo : 인터럽트 번호(0<=IntNo<=255)
Vector : 실행 루틴 번지. 앞에 @를 붙이면 실행루틴 프로시듀어의 번지를 나타냄

위의 두 프로시듀어는 MS-DOS의 INT 21H에서 펑션 35H,

〔화면 2〕 시간변경 확인

```
C:\DOS>DEBUG
-D40:6C                                                      ┌현재 14시
0040:0060                                      CE 2D 0E 00              .-..
0040:0070  00 00 34 12 00 01 F8 F0-14 14 14 14 01 01 01 01   ..4.............
0040:0080  1E 00 3E 00 00 00 00 00-00 00 00 00 00 00 00 00   ..>.............
0040:0090  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0040:00A0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0040:00B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0040:00C0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0040:00D0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0040:00E0  00 00 00 00 00 00 00 00-00 00 00 00              ............
-
                                    잠시후 시간증가
-D40:6C                                                      ┌14시
0040:0060                                      61 2F 0E 00              a/..
0040:0070  00 00 34 12 00 01 F8 F0-14 14 14 14 01 01 01 01   ..4.............
0040:0080  1E 00 3E 00 00 00 00 00-00 00 00 00 00 00 00 00   ..>.............
0040:0090  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0040:00A0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0040:00B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0040:00C0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0040:00D0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
0040:00E0  00 00 00 00 00 00 00 00-00 00 00 00              ............
-
-
-
```

〔그림 4〕 인터럽트 NUM 발생시 처리과정



(a) 평상적인 인터럽트 발생순서

(b) 새로운 루틴에 대한 실행순서

25H를 라이브러리화한 것이다. 다음은 MS-DOS의 펑션전송 인터럽트에서 인터럽트 벡터를 구하는 펑션과 인터럽트 벡터를 세트시키는 펑션을 나타낸 것이다.

```
INT 21H DOS Function 35H : 인터럽트 벡터에서 인터럽트 핸들러 루틴의
                           번지를 구함
호출방법 AH=35H
         AL=인터럽트 번호
리턴 ES : BX=인터럽트 벡터(핸들러 세그먼트 : 오프셋)
INT 21H DOS Function 25H : 인터럽트 벡터를 인터럽트 핸들링 루틴을
                           호출하도록 초기화함
호출방법 AH=25H
         AL=인터럽트 번호
         DS : DX=인터럽트 핸들링 루틴의 새 번지(세그먼트 : 오프셋)
```

인터럽트 키워드 사용법

이것은 파스칼에서 지시자로 사용되는 명령어로 모든 인터럽트 프로시듀어는 다음과 같은 형식을 가져야 한다. 그리고 리스트 1에는 이의 사용예를 나타내었다.

```
Procedure 이름(Flags, CS, IP, AX, BX, CX, DX, SI, DI, DS, ES, BP :
Interrupt;                                     Word) ;
begin
  ⋮
end ;
( 자신의 새로운 루틴(Key_Status, Display_Clock)
```

〔화면 3〕 SK 프로그램 실행 전과 후 (INT 1CH에 들어 있는 루틴 번지 변경)

```
C:\DOS>DEUG
-D0:70              ┌──────── 실행전
0000:0070  33 FF 00 F0 A4 F0 00 F0-22 05 00 00 33 FF 00 F0   3......."...3...
0000:0080  3F 14 2B 02 60 14 2B 02-F4 02 96 0C 2F 03 96 0C   ?.+.W.+...../...
0000:0090  BC 02 96 0C DC 15 2B 02-1F 16 2B 02 66 63 2B 02   ......+...+.fc+.
0000:00A0  45 14 2B 02 9C 06 70 0-45 14 2B 02 45 14 2B 02   E.+...p.E.+.E.+.
0000:00B0  45 14 2B 02 45 14 2B 02-81 02 76 0B 0C 18 70 00   E.+.E.+...v...p.
0000:00C0  EA 46 14 2B 02 14 2B 02-45 14 2B 02 45 14 2B 02   .F.+..+.E.+.E.+.
000:00D0  45 14 2B 02 45 14 2B 02-45 14 2B 02 45 14 2B 02   E.+.E.+.E.+.E.+.
0000:00E0  45 14 2B 02 45 14 2B 02-45 14 2B 02 45 14 2B 02   E.+.E.+.E.+.E.+.
-Q

C:\DOS>9

C:\DOS>DEBUG
-D0:70              ┌──────── 실행후
0000:0070  E0 08 5E 0C A4 F0 00 F0-22 05 00 00 33 FF 00 F0   ..^......"...3...
0000:0080  3F 14 2B 02 72 0E 5E 0C-F4 02 51 26 2F 03 51 26   ?.+.r.^...Q&/.Q&
0000:0090  BC 02 51 26 A9 0E 5E 0C-BF 0E 5E 0C 66 63 2B 02   ..Q&..^...^.fc+.
0000:00A0  FE 09 5E 0C 9C 06 70 00-45 14 2B 02 45 14 2B 02   ..^...p.E.+.E.+.
0000:00B0  45 14 2B 02 45 14 2B 02-81 02 76 0B 0C 18 70 00   E.+.E.+...v...p.
0000:00C0  EA 46 14 2B 02 14 2B 02-45 14 2B 02 45 14 2B 02   .F.+..+.E.+.E.+.
0000:00D0  45 14 2B 02 45 14 2B 02-45 14 2B 02 45 14 2B 02   E.+.E.+.E.+.E.+.
0000:00E0  45 14 2B 02 45 14 2B 02-45 14 2B 02 45 14 2B 02   E.+.E.+.E.+.E.+.
-
```

인터럽트 프로시듀어의 인터럽트를 실행함으로써 모든 레지스터는 자동적으로 저장되고 DS 레지스터를 초기화한다. 즉 리스트 2의 과정이 자동적으로 실행된다. 그리고 인터럽트 프로시듀어 실행이 끝나면서 모든 레지스터를 자동적으로 리스트 3과 같은 과정으로 복원하고 인터럽트 리턴 명령을 만남으로써 인터럽트 프로시듀어 실행이 끝나게 된다.

프로그램 구성

리스트 4에서 보여준 전체 프로그램은 모두 6개의 프로시

〔리스트 1〕 인터럽트 키워드 사용 예

```
Procedure New_Interrupt_1CH(Flags,CS,IP,AX,BX,
                  CX,DX,SI,DI,DS,ES,BP : Word);
Interrupt ;
begin
  Inline( $CD / $60) ;
  Key_Status ;
  Display_Clock
end ;
```

〔리스트 2〕 레지스터 저장

```
PUSH   AX
PUSH   BX
PUSH   CX
PUSH   DX
PUSH   SI
PUSH   DI
PUSH   DS
PUSH   ES
PUSH   BP
MOVE   BP,SP
SUB    SP,LocalSize
MOVE   AX,SEG DATA
MOVE   DS,AX
```

〔리스트 3〕 레지스터 복구

```
MOVE   SP,BP
POP    BP
POP    ES
POP    DS
POP    DI
POP    SI
POP    DX
POP    CX
POP    BX
POP    AX
IRET
```

〔리스트 4〕 전체 리스트

```
Program Interrupt_Handler ;

Uses
  CRT , DOS ;
Const
  BIOS_Area = $0040 ;
  KB_Flag   = $0017 ;
  Caps      = $40   ;
  Num       = $20   ;
  Scroll    = $10   ;
  VRAM      = $B000 ;
  INVERSE   = $70   ;
  NORMAL    = $07   ;
Var
  Old_Interrupt_1CH ,
  Old_Interrupt_60H : Pointer ;
  VBuffer : Array [1..25 , 1..80] Of Integer Absolute
                                                   VRAM:$0000 ;

Procedure Key_Status ;
var
   KB_Status , Attribute : Byte ;
begin
   KB_Status := Mem[BIOS_Area:KB_Flag] ;
   If ( KB_Status And Caps ) <> 0 THEN Attribute := INVERSE
                                  ELSE Attribute := NORMAL ;
      VBuffer[25,66] := (Attribute Shl 8) + Ord('C') ;
      VBuffer[25,67] := (Attribute Shl 8) + Ord('a') ;
      VBuffer[25,68] := (Attribute Shl 8) + Ord('p') ;
      VBuffer[25,69] := (Attribute Shl 8) + Ord('s') ;
   If ( KB_Status And Num ) <> 0 THEN Attribute := INVERSE
                                 ELSE Attribute := NORMAL ;
      VBuffer[25,71] := (Attribute Shl 8) + Ord('N') ;
      VBuffer[25,72] := (Attribute Shl 8) + Ord('u') ;
      VBuffer[25,73] := (Attribute Shl 8) + Ord('m') ;
   If ( KB_Status And Scroll ) <> 0 THEN Attribute := INVERSE
                                    ELSE Attribute := NORMAL ;
      VBuffer[25,75] := (Attribute Shl 8) + Ord('S') ;
      VBuffer[25,76] := (Attribute Shl 8) + Ord('c') ;
      VBuffer[25,77] := (Attribute Shl 8) + Ord('r') ;
      VBuffer[25,78] := (Attribute Shl 8) + Ord('o') ;
      VBuffer[25,79] := (Attribute Shl 8) + Ord('l') ;
      VBuffer[25,80] := (Attribute Shl 8) + Ord('l') ;
end ;

Procedure Display_Clock ;
var
   Hour, Min, Sec, Sec100 : Word ;
begin
   GetTime( Hour, Min, Sec, Sec100 ) ;
   If Hour > 12 then Hour := Hour-12 ;
   VBuffer[1,70] := (INVERSE Shl 8) + ((Hour Div 10) +$30) ;
   VBuffer[1,71] := (INVERSE Shl 8) + ((Hour Mod 10) +$30) ;
   VBuffer[1,73] := (INVERSE Shl 8) + ((Min Div 10) +$30)  ;
   VBuffer[1,74] := (INVERSE Shl 8) + ((Min Mod 10) +$30)  ;
   VBuffer[1,76] := (INVERSE Shl 8) + ((Sec Div 10) +$30)  ;
   VBuffer[1,77] := (INVERSE Shl 8) + ((Sec Mod 10) +$30)  ;
   VBuffer[1,79] := (INVERSE Shl 8) + ((Sec100 Div 10) +$30) ;
   VBuffer[1,80] := (INVERSE Shl 8) + ((Sec100 Mod 10) +$30)
end ;

Procedure New_Interrupt_1CH(Flags,CS,IP,AX,BX,CX,DX,SI,
                                        DI,DS,ES,BP : Word) ;
Interrupt ;
begin
   InLine( $CD / $60 ) ;
   Key_Status ;
   Display_Clock
end ;

Procedure Install ;
begin
   InLine( $FA ) ; { CLI }
   GetIntVec( $1C , Old_Interrupt_1CH ) ;
   GetIntVec( $60 , Old_Interrupt_60H ) ;
   SetIntVec( $60 , Old_Interrupt_1CH ) ;
   SetIntVec( $1C , @New_Interrupt_1CH ) ;
   InLine( $FB )   { STI }
end ;

Procedure UnInstall ;
begin
   InLine( $FA ) ; { CLI }
   SetIntVec( $1C , Old_Interrupt_1CH ) ;
   SetIntVec( $60 , Old_Interrupt_60H ) ;
   InLine( $FB )   { STI }
end ;

Procedure Demo ;
const
   L : Byte = 7 ;
var
   String7 , Count7 : Array[1..7] of Byte ;
   I, J, K, MaxChar : Byte                ;
   Ch               : Char                ;
begin
   TextAttr := NORMAL; Clrscr ;
   FillChar( Count7 , 7 , #0 ) ;
   For I := 1 to 7 do begin
      GotoXY( I , 3 ) ;
      Ch := ReadKey ;
      VBuffer[3,I] := (NORMAL Shl 8) + Ord(Ch) ;
      String7[I] := Ord(Ch)
   End ;
```

```
MaxChar := String7[1] ;
For I := 2 to 7 do
   If ( MaxChar <= String7[I] ) then MaxChar := String7[I] ;
For I := 1 to MaxChar do begin
   For J := 1 to 7 do begin
      If (String7[J] > Count7[J]) then begin
         For K := 4 to 24 do begin
            VBuffer[K,J+L] := (INVERSE Shl 8) + I ;
            Inc( L )
         End ;   {-- Of For K := --}
         L := 7 ;
         Delay( 35 ) ;
            Inc( Count7[J] )
         End   {-- Of If --}
      End   {-- Of For J := --}
   End   {-- Of For I := --}
end ;

Begin
  Install ;
  Demo ;
  Repeat Until KeyPressed ;
  UnInstall
End.
```

듀어로 구성된 간단한 프로그램이다. 각각의 프로시듀어를 살펴보면 다음과 같다.

- **Key_Status**: 현재의 키상태(Caps, Num, Scroll)를 화면 우측 하단에 표시
- **Display_Clock**: 현재 시간을 시, 분, 초, 1/100초로 구성하여 화면 우측 상단에 표시
- **New_Interrupt_1CH**: 키상태 체크와 시간을 표시하는 새로운 인터럽트 1CH 루틴
- **Install**: 벡터 테이블에 인터럽트 1CH, 60H의 새로운 실행 루틴 번지를 기록
- **Uninstall**: 벡터 테이블에 원래의 인터럽트 1CH, 60H의 실행루틴 번지를 기록
- **Demo**: 7개의 문자를 입력하면 각각의 문자에 대해서 아스키 코드값 0부터 해당 문자 코드값까지의 문자를 한자씩 출력시킴

프로그램 설명

처음 Install 프로시듀어를 실행하면 두 개의 인터럽트 벡터를 변경하는데 원래의 번지는 Old_Interrupt_1CH와 Old_Interrupt_60H라는 포인터변수에 저장된다. 원래의 인터럽트 1CH에 새로운 루틴 New_Interrupt_1CH가 세트되고, 인터럽트 60H에는 원래의 인터럽트 1CH가 세트된다. 여기서 인터럽트 60H는 사용자를 위한 인터럽트로 원래의 인터럽트 1CH를 실행하기 위함이다.

다음으로 Demo를 호출하면 Demo가 실행되면서 동시에 1초에 18.2번씩 혹은 0.0549254초에 한번씩 인터럽트 1CH 즉 New_Interrupt_1CH가 실행되어 화면 상단과 하단 우측에 각각 시간과 키상태가 나타난다. 따로 New_Interrupt_1CH를 호출하지도 않았는데 마치 세개의 루틴(Key_Status, Display _Clock, Demo)이 동시에 수행되는 것처럼 보인다. 이것이 바로 인터럽트 핸들러의 묘미인 것이다.

Demo가 끝난 후에도 키상태와 시간은 계속해서 실행하는데 이때 아무키나 누르면 원래의 번지를 세트하고 전체프로그램을 종료시킨다. 여기서 키상태는 세그먼트 : 오프셋 $0040 : $0017의 1바이트를 각 비트조작(And)으로 검출하여 표시한 것이고, 시간은 터보파스칼의 라이브러리를 사용하였다. 프로그램에서 세 개의 InLine문을 사용하였는데 InLine($CD/$60)은 INT 60H와 대응되는 것으로 결과적으로는 원래의 인터럽트 1CH를 실행하는 것이다. InLine($FA)와 InLine($FB)는 인터럽트 변경중 하드웨어 인터럽트가 발생하는 것을 막기 위한 것이다.

맺음말

지금까지 설명되어온 프로그램을 실행시키면 데모프로그램의 실행중에도 시간 표시와 키상태 표시가 정지되지 않고 계속해서 시간을 출력하고 키상태를 체크하여 인버스시키는 것을 보았을 것이다. 이와 같은 현상이 정상적으로 일어난다면 지금까지 설명한 인터럽트 핸들러의 의도를 충분히 알 수 있으리라 본다.

그리고 이 프로그램에서 한가지 주의해야 할 점은 인터럽트 프로시듀어 내에서 도스 펑션 콜을 사용한 후 디스크를 액세스하면 시스템이 중지한다는 것이다. 이러한 경우가 왜 생기는지 독자 여러분이 찾아보길 바라고, 인터럽트 루틴을 사용할 때는 항상 주의하기를 바란다.

또 한가지 이 프로그램에서 아쉬운 것은 시간 관계상 램상주 프로그램까지 발전시키지 못했다는 것이다. 다음에 다시 기회가 주어져 램상주 프로그램을 소개하면 프로그램의 실행과 관계없이 시간표시와 키상태를 항상 표시할 수 있을 것이다.

이 프로그램을 작성하면서 한가지 여담은 실제 필요로 하는 루틴보다 데모프로그램 작성에 더 많은 시간을 소비했다는 것이다. 독자 여러분중 좀더 관심있는 분은 이 두가지 루틴을 여러분의 프로그램에 응용할 수 있을 것이며, 좀더 나아가 독자들이 작성한 램상주 프로그램이 나온다면 이 프로그램이 제시한 의도가 충분히 전해졌다고 할 수 있다. 독자 여러분의 건투를 빈다. PC

터보 베이직의 통합 환경

박연진 / 여주컴퓨터학원 강사

연재를 시작하며

베이직은 확실히 배우기 쉬운 언어이다. 그러므로 컴퓨터가 없더라도 베이직 책이 있다면 혼자서도 그럭저럭 배워 나갈 수도 있다. 그런 까닭에 베이직을 사용하는 사람들은 대부분 전문적인 전산학을 배운 사람들보다는 비전공인이 훨씬 더 많다. 본지에도 베이직 강좌가 연재되고 있는 만큼 초보자에게 있어서 베이직이 차지하는 비중은 매우 크다고 볼 수 있다.

그러나 어설픈 무당이 사람 잡는다는 속담이 있듯이 형식과 격식을 제대로 갖추지 못한 책을 가지고 공부한 사람들은 아주 나쁜 버릇이 들기 쉽다. 보기 좋은 떡이 맛이 더 좋다는 말처럼 좋은 프로그램은 우선 보기에도 좋아야 한다. 좋은 떡을 만들기 위해서는 만드는 사람의 솜씨도 좋아야 하겠지만 먼저 좋은 쌀을 골라야 할 것이다. 밀기울을 가지고서 어찌 눈처럼 하얀 백설기를 만들 수 있을 것인가? 좋은 프로그램을 작성하려면 우선 좋은 언어를 선택해야 한다.

본 강좌는 터보 베이직이란 좋은 언어를 가지고 기초부터 고급과정에 이르기까지 연재될 예정이다.

베이직이란 무엇인가?

베이직(BASIC)은 1965년 다트무스(Dartmouth) 대학의 존 케머니(John. G.Kermerny)와 쿠르츠(Kurtz)에 의해 고안된 고급언어이다. 당시 이 교수들이 BASIC을 설계할 당시에는 전문적인 프로그램용 언어로 사용할 목적은 없었고, 시분할 장치(TSS)가 가능한 대형 컴퓨터에서 학생들에게 컴퓨터의 동작원리를 설명하기 위한 간이 언어로 출발하였다. 그래서 언어의 이름도 'Beginner's All-purpose Symbolic Instruction Code'의 첫 문자를 조합해서 붙인 것이다. 이것을 구태여 번역하자면 '초보자들이 모든 방면에서 사용 가능한 명령 코드'라고나 할까?

이것은 포트란(FORTRAN)을 개량해서 만든 것으로, 비록 기능은 빈약하지만 초보자가 배우기 쉽도록 단순화하였기 때문에 쉽게 배울 수 있는 매력을 가지고 있었다. 또한 베이직은 다른 고급언어에 비해 메모리를 최소로 사용한다는 특성 때문에 소형 기종에 이식하기가 쉬웠다.

1971년 미국의 인텔사에 의해 8비트 마이크로 프로세서가 탄생했고, 이어 최초의 퍼스널 컴퓨터라고 할 수 있는 알테어 8080이 등장했다. 이 컴퓨터는 베이직 인터프리터가 8KB, 사용자 메모리가 8KB인 요즈음의 전자 오락기보다도 못한 메모리를 가지고 있었으나 부동 소숫점의 연산이 가능하고, 삼각함수도 취급할 수 있는 그때 당시로는 경이적인 작품이었다.

이 컴퓨터의 인터프리터(명령이나 프로그램을 해석하여 실행시키는 컴퓨터 내부의 프로그램)는 당시 하버드 대학생이었던 빌 게이츠와 앨린에 의해 작성되었으며, 그 후 그들은 다니던 대학도 때려치우고 마이크로소프트(Microsoft) 사를 설립하여 세계 굴지의 프로그램 하우스로 성장시켰다. (현재 독자들이 사용하고 있는 컴퓨터의 MS-DOS나 GW-BASIC, 멀티 플랜 등이 모두 마이크로소프트사의 제품이다.)

그 후 1977년에 이르러 저 유명한 애플Ⅱ가 탄생하면서부터 퍼스널 컴퓨터가 본격적으로 보급되기 시작하였고, 여기 저기서 퍼스널 컴퓨터를 생산하는 업체들이 우후죽순격으로 나타났다. 각 메이커들은 세일즈의 한 전략으로 타사보다도 더 다양한 명령어를 구사할 수 있는 베이직 인터프리터를 내세웠으며, 당시의 포트란이나 코볼 등의 고급언어에서는 볼 수 없는 그래픽과 사운드 기능을 강화하였다. 따라서 베이직은 초기의 형태에서 빠른 속도로 발전을 거듭하여 왔으며, 어느 언어보다 급속하게 확산되었다.

그러한 결과 베이직은 아직까지 어떤 표준이 없으며, 미국 표준협회 기관인 ANSI가 표준 베이직을 제정했지만 이것을 고수하는 업체는 아직까지도 없고, 수십가지의 베이직이 존재하게 되었다. 컴퓨터의 대부라고 할 수 있는 IBM에서 개인용 컴퓨터 시장에 눈을 돌리면서 16비트 컴퓨터를 생산할 무렵 IBM사는 그 운영체제의 전반을 빌 게이츠가 회장으로 있는 MS사에 의뢰하였고, MS사는 MS-DOS와 함께 자사가 개발한 MS-BASIC을 개량하여 GW-BASIC이라는 이름으로 사용자들에게 보급하기에 이르렀다.

IBM-PC는 오늘날 우리나라에서는 행정 전산망의 표준 기종으로 자리를 굳히게 되었고, 따라서 IBM-PC의 사용자들은 자연스럽게 GW-BASIC을 접할 수 있게 된 것이다.

베이직의 단점

베이직은 배우기 쉽고 프로그램을 작성하기 쉬우며 프로그램의 개발 비용이 적게 든다. 또한 비교적 하드웨어와 밀접한 그래픽과 사운드 명령이 내장되어 있어서 아기자기한 프로그램을 만들 수가 있다. 또한 인터럽트가 다양하므로 다른 언어에서는 비교적 까다로운 부분도 쉽게 구현될 수 있다.

그러나 베이직은 이런 장점이 있음에도 불구하고 많은 단점을 가지고 있다. 우선 인터프리터 방식이기 때문에 프로그램의 수행속도가 컴파일형의 언어인 코볼이나 포트란보다 현저하게 떨어진다. 그러나 수행속도가 늦다는 것은 요즈음에 와서는 별로 단점으로 지적되지 않는다. XT 대신 AT를 쓴다든지 클럭 주파수가 높은 CPU를 사용하는 등의 하드웨어적인 방법으로도 스피드업을 꾀할 수 있을 뿐더러 베이직 컴파일러가 개발되어 있으므로 일단 인터프리터 방식에서 프로그램을 작성한 후 컴파일러로 컴파일시키면 수행속도가 놀랍게 빨라질 수 있기 때문이다.

전문가들이 베이직의 단점을 지적하는 것은 이런 이유 때문이 아니고 다음의 사항 때문이다.

① 베이직은 번거로운 행번호를 기술해야 한다.
② 베이직은 내장 에디터의 기능이 빈약하다.
③ 베이직에서 다룰 수 있는 메모리의 한계가 64KB 뿐이다.
④ 구조적 프로그래밍 환경을 갖추지 못했다.

이들 중 ①과 ②는 그런대로 넘어갈 수도 있는 문제이지만 ③과 ④는 대단히 심각한 문제를 가져온다. 베이직에서 다룰 수 있는 메모리의 한계가 64KB 뿐이라는 것은 사용자가 작성한 프로그램과 그 프로그램에서 사용하는 변수, 배열 등을 합한 수치이므로 규모가 큰 프로그램이나 대량의 데이터를 취급하는 프로그램을 작성할 경우에 커다란 벽에 부딪치게 된다. 물론 오버레이기법을 쓰거나 가상 메모리를 이용하면 전혀 불가능한 것은 아니지만 흔히 하는 말로 남들은 쉽게 하는 것을 혼자서 어렵게 하는 결과를 가져온다.

또한 구조적 환경을 갖추지 못했다는 것 역시 결정적인 결점중의 하나이다.

프로그램은 작성하는데 적은 노력이 들어야 하겠지만 그것을 보수하는데도 적은 노력과 비용이 들어야 한다.

구조적 환경을 갖추지 못한 언어에서는 어떤 문제를 처리할 경우에 복잡한 제어구조를 사용해야 하므로 프로그램의 논리를 추적하기 곤란하게 만든다.

제어문의 하나인 GOTO문은 그것이 프로그램에 많이 나타날수록 프로그램의 추적은 그것의 몇갑절에 비례해서 어려워진다. 그러므로 프로그램의 작성시 GOTO문은 되도록 사용을 억제해야 하며, 가능하다면 전혀 사용하지 않아야 한다. '프로그래머의 자질은 그 사람이 작성하는 프로그램에서 나타나는 GOTO문의 횟수에 반비례한다'라는 격언이 있듯이 GOTO문은 절대 해롭다.

그러나 구조적인 환경을 갖지 못한 언어에서는 GOTO문을 사용하지 않으면 프로그램을 작성하지 못하는 경우가 비일비재하다. 베이직 역시 이 범주에 속한 언어이므로 어쩔 수 없이 GOTO문을 사용해야만 한다.

그러므로 베이직으로 프로그램을 작성하는 경우 프로그램을 작성하는 사람은 쉬울런지 모르지만 프로그램을 보수하는데는 많은 시간과 노력이 들게 된다.

터보베이직의 등장

미국에서는 벌써부터 베이직의 이런 단점들을 개선하기 위해 많은 노력을 기울여 왔다. 트루베이직(True BASIC)이나 슈퍼베이직(Super-Basic) 등이 등장하여 구조적 프로그래밍 환경을 지원하기 시작한 것이다. 여기에 자극을 받은 소프트업계의 큰 손격인 MS사가 퀵베이직(Quick Basic)을 발표하여 베이직을 전혀 새로운 모습으로 바꾸어 놓았다.

퀵베이직은 버전이 1.0부터 4.xx까지 발전되어 왔는데 퀵베이직이 2.0이 발표될 무렵 사이드 킥을 비롯하여 터보 파스칼, 터보 C 등의 터보 시리즈로 이름을 날리던 볼랜드사에서 퀵베이직보다 더 완벽한 환경을 갖춘 터보베이직을 발표하였다.

발표 당시 터보 베이직은 확실히 퀵베이직보다 한 수 위였다. 퀵베이직에는 지원하지 못한 환경을 터보 베이직은 거의 무리없이 지원하였으며 터보라는 명성에 조금도 부끄러움이 없는 작품이라는 평가를 받기에 이르렀다. 여기에 MS사도 자사의 퀵베이직을 더욱 발전시켜서 터보를 능가하는 퀵베이직 4.xx을 발표하여 오늘에 이르게 된 것이다. 이제는 볼랜드사에서 터보 베이직 2.0을 내놓지 않는한 우위는 퀵베이직으로 넘어가게 되었다.

그러나 MS사가 1.0~4.xx까지 버전을 개량하면서 여러번의 시행착오를 거듭해 온 반면 볼랜드사는 단 한번에 대단한 걸작을 만들어 냈다는 사실은 많은 사람들에게 깊은 신뢰감은 주고 있으며, 터보 시리즈라는 단 한가지 사실 때문에 터보 베이직을 사용하는 사람들도 많다. 만약 터보 베이직 2.0이 만들어진다면 분명히 전세는 다시 역전될 것이다.

GW-BASIC과의 차이점

그러면 이런 베이직들이 GW-BASIC

과 무엇이 다른지를 알아보기로 하자. 우선은 이들 베이직은 행번호가 없다. 물론 행번호를 붙여도 되지만 번거로운 행번호를 구태여 붙이려는 사람은 없을 것이다. 또한 이들은 컴파일형의 언어이므로 프로그램의 실행속도가 대단히 빠르다.

그러나 무엇보다도 중요한 점은 이들은 완벽한 구조적 언어의 특성을 가지고 있다는 점이다. 그러므로 터보 베이직이나 퀵베이직을 쓰는 한 GOTO문을 전혀 사용하지 않고도 프로그램을 작성하는 것이 가능하다. 또한 완벽한 블록구조를 지원하므로 프로그램을 작성·분석하기가 쉬우며, 모듈별 프로그래밍이 가능해진다.

이들 베이직은 통합 환경내에서 동작하므로 프로그램을 작성하여 프로그램의 실행은 물론 컴파일이 된다. 그러므로 포트란이나 코볼의 경우처럼 컴파일한 후 따로 링크할 필요가 없으므로 컴파일하는 시간이 매우 짧다. 또한 64KB라는 메모리의 제한을 받지 않으므로 커다란 프로그램, 거대한 데이터를 다루는 프로그램을 작성하는 것이 가능하다.

또한 이들 베이직은 매우 강력한 기능을 갖는 에디터를 제공한다. 이 에디터는 상하좌우 스크롤은 물론 블록 단위의 편집과 검색 치환의 기능이 있으므로 프로그램을 아주 쉽게 작성할 수 있다. 단지, GW-BASIC에서 지원하는 펑션키의 기능은 없으므로 명령어를 일일이 기술해야 한다는 불편은 있으나 요즈음은 펑션키를 사용자가 정의해주도록 하는 프로그램들이 나와 있으므로 이런 기능을 활용하면 GW-BASIC 보다 훨씬 더 다양하게 펑션키를 활용할 수 있다.

또한 강력한 SHELL명령을 지원하므로 프로그램 수행중에 DOS로 빠져나와 다른 작업을 할 수도 있으며, 작업이 끝난 후 다시 예전의 프로그램으로 돌아가서 프로그램을 계속 수행시킬 수도 있다. 이런 기능을 이용하면 베이직의 서브루틴으로 코볼이나 파스칼, C 등으로 작성한 프로그램을 활용하는 것이 가능하다.

퀵베이직과의 차이점

터보 베이직이나 퀵베이직은 GW-BASIC을 바탕으로 하여 만들어졌기 때문에 GW-BASIC으로 작성된 프로그램은 거의 대부분 제대로 수행된다. 이들 베이직은 GW-BASIC에는 없는 새로운 기능을 많이 강화하였는데 각사가 서로 독자적으로 개발하였기 때문에 확장된 부분에 있어 호환성이 없는 부분이 더러 있다.

그러나 너무 한쪽에만 치우친 명령을 사용하지만 않는다면 서로 호환성 있는 프로그램을 작성하는 것이 가능하다. 어셈블리 인터페이스의 처리는 서로 다르며, 터보 베이직이 월등히 앞선다. 만일 어셈블리 루틴을 자주 사용하는 경우에는 터보 베이직을 사용하는 편이 훨씬 더 유리하다. 두 베이직의 차이점을 나열하는 것만으로도 몇회 분의 분량이 되므로 세부적인 차이는 그때그때 설명하기로 하자.

이 강좌에서는 터보 베이직을 위주로 하여 설명을 하되, 퀵베이직 사용자를 위해서는 따로 수정해야 할 부분을 게재할 예정이므로 퀵베이직 사용자들은 그런 부분을 참고하기 바란다. 이 강좌는 매뉴얼적인 성격을 배제하고 알고리즘에 중점을 두어 설명할 것이므로 터보 베이직의 매뉴얼을 한권쯤 사서 보는 편이 좋을 것이다.

터보 베이직의 설치

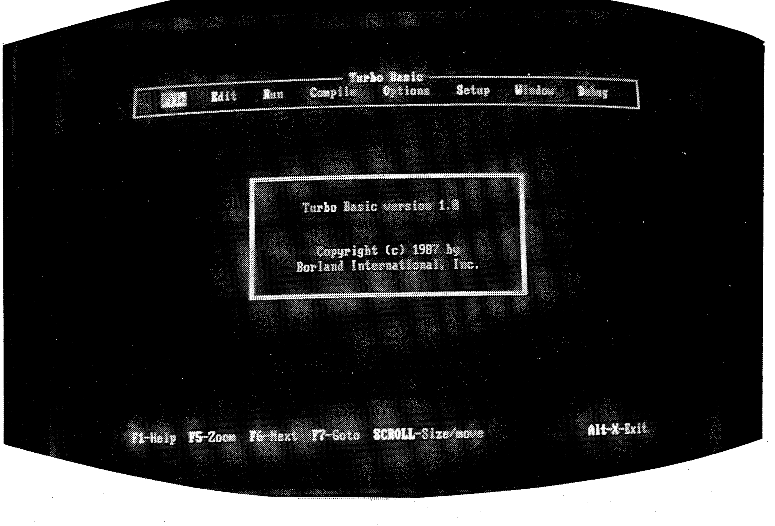
터보 베이직은 2장의 디스켓에 담겨 보급된다. 이 디스켓의 #2는 예제 프로그램이 수록되어 있는 것이므로 없어도 무방하다. #1의 내용은 다음과 같다.

- README.COM : README의 내용을 화면에 표시하는 프로그램
- TB.EXE : 터보 베이직 메인 화일
- TBHELP.TBH : 도움말 메시지
- TBINST.COM : TB.EXE의 인스톨 화일
- TBCONFIG.TB : 환경 화일(사용자의 지정에 의해 만들어진다.)
- README : 터보 베이직의 실행 방법 등을 기술한 텍스트 화일

터보 베이직은 다음 명령에 의해 실행된다.

```
A>TB ↵
```

〔화면 1〕 터보베이직의 초기화면



그러면 잠시 후 **화면 1**과 같은 상태가 된다. 여기서 아무 키나 누르면 **화면 2**와 같이 바뀐다. 터보 베이직은 4개의 윈도우를 가지고 있으며 각각의 윈도우는 그 크기와 속성을 사용자가 자유로이 지정하여 변경시킬 수 있다.

윈도우간의 이동은 〈F6〉키로 하며, 해당하는 윈도우는 밝게 하이라이트된다. 각 윈도우의 〈F5〉를 누르면 화면 전체로 확대되며 다시 한번 〈F5〉를 누르면 원래의 크기로 돌아온다. (이렇게 화면을 확대, 축소하는 기능을 Zoom in, Zoom out이라고 부른다)

〈Scoroll Lock〉키가 눌려져 있으면 각각의 윈도우는 커서키에 의해 이동을 하게 되므로 사용자가 자신의 취향에 맞도록 윈도우를 배치하는 것이 가능하다. 예를 들어 에디트 윈도우를 화면의 우측 상단으로 이동하고 싶으면 다음과 같은 절차를 밟으면 된다.

① 〈F6〉키를 눌러 에디트 윈도우를 선택한다.

② 〈Scoroll Lock〉키를 켜고 커서 키로 에디트 윈도우를 이동한다. 그러면 그 자리에 있던 윈도우 위에 에디트 윈도우가 겹쳐진다.

③ 〈F6〉키를 눌러 에디트 윈도우 밑에 들어가 있는 윈도우를 선택한다.

④ 그 윈도우를 빈 자리로 끌어내린다.

또, 〈Scroll Lock〉키가 켜진 상태에서는 〈Shift〉키를 누른 채로 커서 키를 누르면 그 방향으로 윈도우가 커지거나 작아지게 된다. 에디트 윈도우와 런 윈도우는 최대한으로 크게하든지 아예 Zoom in(〈F6〉)하여 두고, 다른 윈도우는 되도록 작게 설정하는 것이 좋다. 이렇게 설정된 윈도우의 속성은 TB-CONFIG. TB에 수록할 수 있으며 수록된 후에는 TB.EXE를 실행할 때마다 설정된 상태로 세팅된다.

메인 메뉴

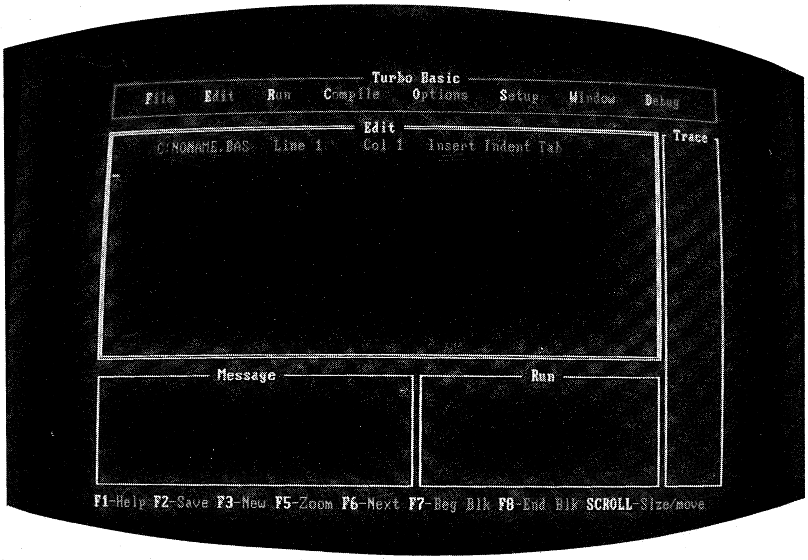
화면 2에서 보듯이 메인 메뉴(Main Menu)는 8개의 항목으로 되어 있으며, 각각의 항목을 선택하면 그 항목의 메뉴가 그자리에 나타난다. 이와 같이 메뉴가 아래로 나무가지처럼 늘어지며 전개되는 방식을 풀다운(Full Down) 메뉴라고 한다. 메인 메뉴에서 항목의 선택은 커서 키로 원하는 항목을 선택하거나 첫문자를 입력하면 된다.

(1) 화일

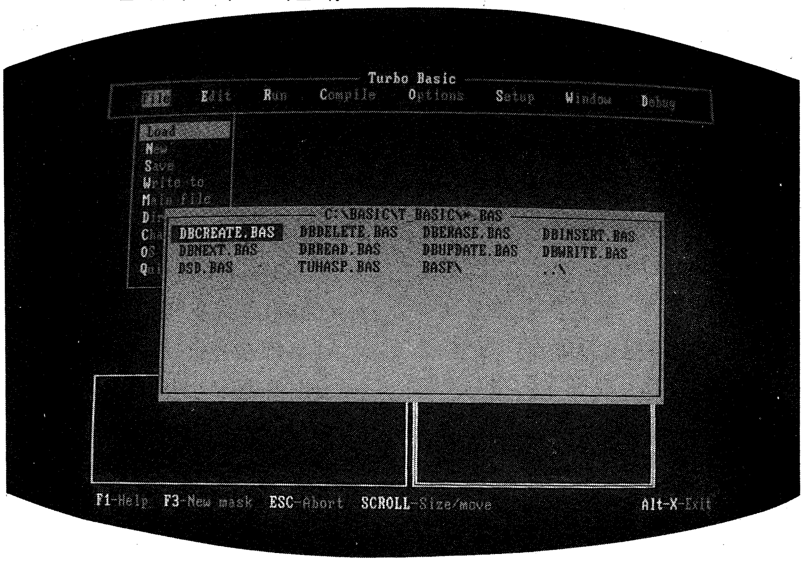
이것을 선택하면 다음과 같은 항목이 나타난다.

- **LOAD**: 디스크의 화일을 로드한다. GW-BASIC으로 작성된 프로그램을 로드하려는 경우에는 그 프로그램이 GW-BASIC의 SAVE"화일명", A와 같이 아스키 형식으로 세이브한 화일만 로드할 수 있다. 이 항목을 선택하면 화일명을 물어오는데 화일명을 입력하지 않고 드라이브 번호만 입력하면 그 드라이브에 수

〔화면 2〕 터보베이직의 윈도우



〔화면 3〕 화일명을 입력하지 않고 키를 치면 커런트 디렉토리에서 확장자가 . BAS인 것이 모두 표시된다.



록된 확장자가 '.BAS'인 화일 목록을 표시하므로 커서를 원하는 이름에 일치 시킨후 ↵ 를 치면 화일명을 일일이 입력하지 않아도 된다(화면 3 참조).

- **NEW**: 현재 편집중인 프로그램을 지운다. 그 프로그램이 아직 저장되지 않은 경우에는 저장할 것인지를 물어온다. 이 때는 Y나 N으로 대답한다.
- **Save**: 편집중인 프로그램을 디스크에 저장한다. 화일 이름을 지정하지 않은 경우 터보 베이직은 NONAME.BAS라고 가정하는데 이것은 이 항목에서 새로운 이름을 주어 디스크에 저장하면 자동으로 바뀌진다.
- **Write to**: 현재 편집중인 프로그램을 다른 이름으로 디스크에 저장한다. 터보 베이직은 새로운 이름으로 물어오므로 새로운 이름을 지정하면 된다.
- **Main file**: 프로그램을 여러 부분으로 나누어 작성하는 경우에 가장 핵심이 되는 화일이름을 기술한다. 터보 베이직은 프로그램의 실행 및 컴파일시 에러가 발생하면 메인 화일에 등록된 프로그램을 로드한 후 편집상태가 된다. 특별히 지정하지 않아도 된다.
- **Directory**: 디렉토리 및 서브 디렉토리의 화일목록을 표시한다.
- **Change dir**: 커런트 디렉토리를 바꾼다.
- **OS shell**: 터보 베이직에서 잠시 DOS로 간다. 돌아올 때는 DOS상태에서 EXIT를 입력하면 된다.
- **Quit**: 터보 베이직을 끝내고 DOS로 돌아온다. 아직 편집중인 프로그램이 저장되지 않는 상태에서 끝낼 경우에는 저장할 것이냐를 물어오는데 Y나 N으로 대답한다.

터보 베이직에서는 〈Esc〉키가 입력의 취소로 쓰인다. 그러므로 〈Esc〉키를 입력하면 직전의 메뉴로 돌아가게 된다.

(2) 편집

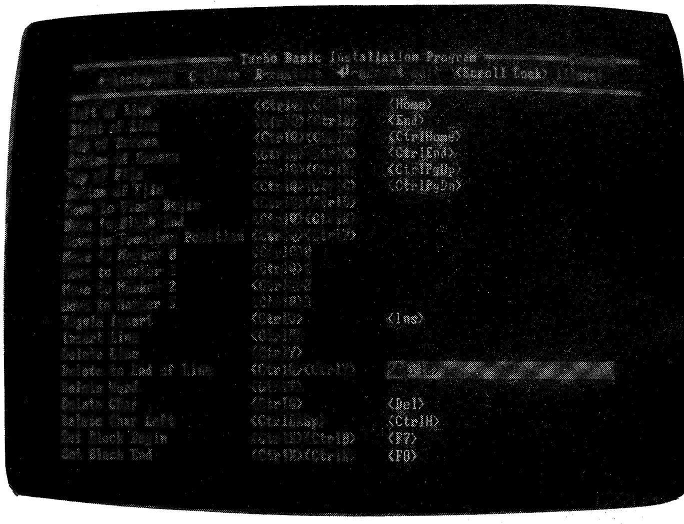
편집모드(Edit Mode)로 들어간다. 이 모드에서 사용하는 편집키는 다음과 같다.

- **F1**: 도움말을 호출한다.
- **F2**: 편집중인 프로그램을 저장한다.
- **F3**: 새로운 화일을 만든다.
- **F5**: 에디트 화면이나 런 화면을 Zoom in / Zoom out한다.
- **Esc**: 메인 메뉴로 복귀
- **Alt-C**: 프로그램의 컴파일(Compile)
- **Alt-R**: 프로그램의 실행(Run)
- **Alt-X**: 터보 베이직을 마치고 DOS로 복귀
- **Alt-F**: File 메뉴로 이동
- **Alt-O**: Option 메뉴로 이동
- **Alt-D**: Debug 메뉴로 이동
- **Alt-W**: Window 메뉴로 이동
- **커서이동키**
  - **Ctrl-A**: 한 단어 좌측으로 이동
  - **Ctrl-F**: 한단어 우측으로 이동
  - **Ctrl-PgUp**: 프로그램의 선두로 이동
  - **Ctrl-PgDn**: 프로그램의 끝으로 이동
  - **Home**: 그 라인의 선두로 이동
  - **End**: 그 라인의 끝으로 이동
- **블록 명령키**
  - **F7**: 블록의 시작 지정
  - **F8**: 블록의 끝 지정
  - **Ctrl-K-C**: 커서의 위치에 블록을 복사
  - **Ctrl-K-V**: 커서의 위치에 블록을 이동
  - **Ctrl-K-R**: 커서의 위치에 블록을 디스크에서 읽어낸다.
  - **Ctrl-K-W**: 블록을 디스크에 저장
  - **Ctrl-Q-B**: 커서를 블록의 선두로 이동
  - **Ctrl-Q-K**: 커서를 블록의 끝으로 이동
  - **Ctrl-Y**: 블록을 삭제
  - **Ctrl-P**: 블록을 프린터로 출력(지정된 블록이 없는 경우에는 프로그램 전체를 프린터로 출력)
- **Ctrl-Y**: 커서 위치에 1줄을 삭제
- **Ctrl-N**: 커서 위치에 1줄을 삽입
- **Ctrl-Q-Ctrl-Y**: 커서 이하 1줄 삭제
- **Ctrl-K-**: 커서의 위치를 기억시킨다. (0~3까지 4군데 설정 가능)
- **Ctrl-Q-**: 위에서 설정해 둔 위치로 커서를 이동시킨다. 이를테면 〈Ctrl-K-3〉으로 커서의 위치를 기억시켜 두었을 때 〈Ctrl-Q-3〉을 입력하면 그 위치로 커서가 이동한다.
- **Ctrl-Q-L**: 잘못 지운 1줄을 복구
- **Ctrl-Q-F**: 문자열의 검색
- **Ctrl-Q-A**: 문자열의 치환
- **Ctrl-O-T**: 탭(Tab) 기능의 설정 해제
- **Ctrl-O-I**: Indent의 설정 및 해제
- **Ctrl-P-**: 콘트롤 문자의 입력(A:1, B:2, C:3…). 예를 들어 아스키코드 3번 문자를 입력하려는 경우에는 Ctrl-P-C를 입력한다. 입력할 당시에는 C자가 반전되어 표시되지만 프로그램을 실행하면 아스키코드 3번 문자가 화면에 출력된다.

위의 편집키는 워드스타(Word star)의 편집키의 기능을 따온 것이다. 터보 파스칼, 터보 C 등의 볼랜드사의 제품은 한결같이 워드스타식의 편집키를 사용한다. 그리고 이 편집키는 사용자가 자유로이 정의하는 것이 허용된다. 이를테면 커서 이하 1줄 삭제 키는 〈Ctrl-Q-Ctrl-Y〉인데 이것을 〈Ctrl-E〉로 바꾼다든지하는 것을 허용한다. 이것은 터보 베이직의 디스크에 있는 TBINST.COM에 있는 기능이며 DOS에서 TBINST.COM을 실행하면 되는데 여기서 'Edit command'를 선택하면 화면 4가 나온다. 이때 수정하려는 키의 항목을 선택한 다음에 새로운 키를 입력한다. 작업이 끝났으면 〈Esc〉를 입력하여 빠져나오게 된다.

이때 이미 존재하는 기능키를 입력한 경우에는 경고 메시지를 출력하고 사용자가 정의한 키 항목으로 커서를 이동시키며, 새로이 정의해 줄 것을 요구한다. 이때 다시 ↵ 를 입력하면 사용자가 정의한 키를 받아들이게 되며, 이미 존재하는 편집키의 기능은 없어진다.

〔화면 4〕 TBINST.COM으로 사용자가 편집키를 편집하는 과정



터보 베이직은 탭의 설정 및 해제가 가능하다. 탭이 설정된 경우에는 〈Tab〉 키를 누르면 다음의 탭존(Tab Zone)으로 커서를 이동시킨다. 그러나 탭이 설정되어 있지 않으면 바로 위에 있는 줄의 상태에 따라 이동하는 거리가 달라진다. 이때는 윗줄의 공백이 아닌 문자가 위치하는 컬럼으로 커서를 이동한다. 이런 기능은 어떤 표를 만든다거나 프로그램의 줄을 맞추어 보기좋게 한다든지 할 때 유용한 기능이다.

터보 베이직은 단짓기(프로그램을 들쑥 날쑥하게 작성하는 것)를 하기 위하여 〈Ctrl-O-I〉로 단짓기(Indent)의 기능을 ON / OFF시킬 수 있다. 이 기능이 ON으로 되어 있으면 한 줄을 입력한 후 키를 치면 커서는 다음 줄의 맨 좌측 컬럼으로 이동하는 것이 아니라 윗 줄의 첫 문자 위치로 커서를 이동하게 된다.

이 기능은 프로그램을 단짓기할 때 매우 유용하다.

Indent의 기능과 탭의 설정 및 해제는 편집모드에서 할 수도 있지만 TBINST. COM에서 설정할 수도 있다. TBINST.COM에서 설정해 두면 매번 부팅할 때마다 사용자가 일일이 지정하는 불편을 없앨 수 있다.

(3) 실행

현재 작성중인 프로그램을 실행한다. 프로그램은 GW-BASIC과는 달리 실행하기 전에 먼저 컴파일 과정을 거치게되는데 컴파일시 에러가 발생하거나 프로그램의 수행중에 에러가 발생하면 자동적으로 편집 모드로 돌아와 에러가 발생한 행에 커서를 이동시키게 된다.

터보 베이직은 편집용 화면과 실행용 화면이 따로 있으므로 에디터상에서 〈F6〉키를 눌러 런 윈도우를 선택하면 직전의 실행 화면을 볼 수 있다.

(4) 옵션

프로그램의 실행이나 컴파일 중에 필요한 옵션을 지정한다.

1. Compile to

컴파일 옵션을 지정한다. 이 항목을 지정하면 다음과 같은 옵션이 나타나는데 이때 커서나 키보드로 적절히 지정하면 된다.

- Memory: 컴파일을 메모리에서 한다.
- Exe file: 지정한 디스크에서 컴파일한다.

  디스크의 지정은 다음에 설명할 셋업 항목에서 한다. DOS상에서 실행 가능한 화일을 만든다.
- Chain file: 체인화일을 만든다. 체인화일은 단독으로 수행될 수는 없으나 Exe 화일에서 CHAIN, RUN 등의 명령으로 실행시키는 것이 가능하다. 체인화일은 Exe 화일에 비해서 컴파일된 오브젝트의 크기가 훨씬 작다. 그러므로 프로그램을 여러개의 모듈로 작성할 경우 메인 프로그램은 Exe 화일로, 서브 프로그램은 체인화일로 컴파일하면 디스크의 스페이스를 대폭 줄일 수 있게 된다.

2. 8087 requird 〈OFF/ON〉

수치 연산용 보조 프로세서인 8087을 탑재한 기종에서는 이 옵션을 ON으로 해두면 처리 속도가 빨라진다.

3. Keyboard break 〈ON/OFF〉

〈Ctrl〉-〈Break〉을 눌러서 프로그램을 강제로 중단하는 기능을 설정할 수 있다. 이 옵션을 OFF로 해두면 프로그램의 실행 속도가 빨라진다.

4. Bounds 〈ON/OFF〉

배열 첨자의 범위가 허용된 한계를 넘어섰는지 여부를 검사할 수 있다. 이 기능에 대해서는 추후 자세하게 설명된다.

5. Stack test 〈ON/OFF〉

스택의 크기가 충분한지의 여부를 체크할 수 있다.

6. Parameter line: (추후 설명함)

7. Metastatments: (추후 설명함)

이들 옵션 중 초심자의 경우는 (2)만 OFF로 해두고 나머지는 ON으로 해두면 된다.

(5) 셋업

이 옵션은 터보 베이직의 제반 환경을 설정한다.

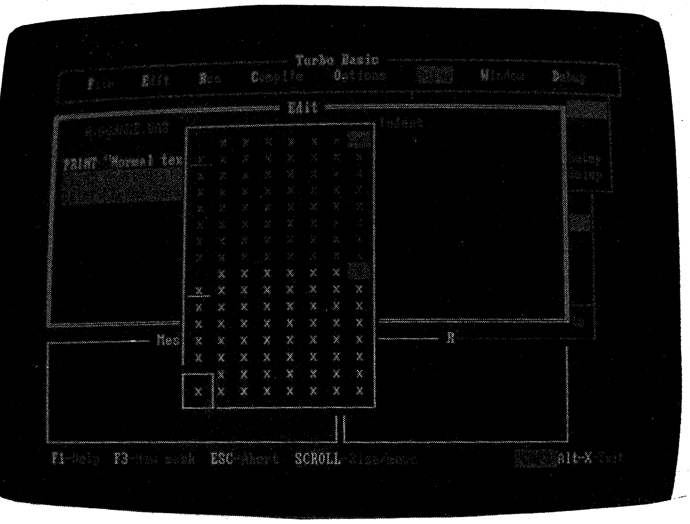
1. Colors

각 윈도우의 테두리(Border), 텍스트, 배경색, 메시지 등의 컬러를 설정 및 변경한다(화면 5 참조).

2. Directories

프로그램을 컴파일할 디렉토리, 또는 프로그램을 불러들어올 디렉토리,

〔화면 5〕 Window 메뉴에서 각 윈도우의 컬러를 선택하는 모습



TB.EXE와 도움말 메시지가 있는 디렉토리 등을 설정한다.

3. Save Option/Window/Setup

위에서 설정한 모든 사항들에 관한 정보를 디스크에 수록한다. 터보 베이직은 TBCONFIG.TB라는 화일명으로 수록하며, 일단 수록된 정보는 터보 베이직이 부트될 때 불러져 앞에서 설정한 사항대로 셋업된다.

4. Load Option/Window/Setup

전에 지정해 둔 옵션 등을 디스크에서 불러낸다. 이 옵션을 선택하면 지금까지 설정한 옵션을 모두 무시하고 예전에 설정해 둔 사항대로 셋업된다.

(6) 윈도우

터보 베이직에는 4개의 윈도우가 있다. 이들 윈도우에 우선 순위를 부여한다든지 하는 항목이다.

(7) 디버그

● Trace 〈OFF/ON〉

프로그램의 트레이스를 지정할 수 있다. 이 옵션이 ON으로 되어있는 경우에는 프로그램을 한 스텝 수행한 후 정지한다. 〈Alt-F10〉을 누르면 다음 스텝을 실행한 후 정지하고 〈Alt-F9〉를 누르면 정지하지 않고 계속 실행한다. 이 때 트레이스 윈도우에는 현재 수행중인 프로그램의 행번호나 레이블이 표시된다. 만약 프로그램 중에 행번호나 레이블이 없다면 트레이스 윈도우에는 아무것도 나타나지 않으며 트레이스도 수행되지 않는다. 그러므로 이 옵션을 설정하기 전에 트레이스할 부분에 행번호나 레이블을 임시로 붙여두어야 한다.

● Run-time error

프로그램 카운터를 지정하여 그 곳의 위치를 찾는다. 이 옵션을 선택하면 'Enter PC : '라는 메시지가 나오게 된다. 터보 베이직의 에디터상에서 프로그램을 실행하는 경우 에러가 발생하면 에러가 발생한 위치로 커서를 이동시켜 주지만 프로그램을 디스크에 컴파일하여 DOS상에서 실행할 경우 에러가 발생하면 'Error 5 Illegal function call at prm-ctr : 29'와 같은 에러 메시지를 출력하고 DOS로 돌아가게 된다. 이 메시지는 '함수의 사용법이 틀렸으며(에러 코드 5번) 그 위치는 처음부터 시작해서 29번째 바이트이다'라는 의미이다. 이렇게 런타임 에러가 발생했을 경우에는 터보 베이직에서 컴파일하기 전의 프로그램을 로드한 후 이 옵션의 'Enter PC : '라는 메시지에 위의 29를 입력하면 에러가 발생한 위치를 찾아준다.

프로그램이란?

프로그램이란 컴퓨터가 처리해야 할 절차를 컴퓨터 언어를 사용해서 순서대로 나열한 일련의 문장을 말하며, 프로그램을 작성하는 사람을 프로그래머라고 부른다. 또 프로그램을 작성하는 것을 프로그래밍이라고 한다.

특별히 조치를 해두지 않는 한 프로그램은 항상 맨 처음부터 순서대로 실행되며, 더 이상 실행할 문장이 없으면 실행을 종료하게 된다. 컴퓨터는 인간이 시키지 않는 일은 절대로 하지 않는다. 또한 컴퓨터는 비판 능력이 없기 때문에 문법에 맞는 명령이라면 그 식의 결과가 맞든 틀리든 수행하게 된다. 그러므로 식을 잘못 기술한 경우에는 전혀 엉뚱한 결과가 나타나며, 그에 대한 책임은 전적으로 프로그래머에게 있다.

프로그램의 작성과 저장 및 실행

먼저 DOS 디스크를 A: 드라이브에 넣고 컴퓨터를 켠 다음 다음과 같이 입력한다. 이때 B: 드라이브는 프로그램을 저장할 포맷된 공디스크를 넣어둔다.

```
A>TB ↲
```

그러면 잠시 후 화면 1과 같은 상태가 된다.

여기서 아무키나 누르면 화면 2의 상태가 되는데 여기서 프로그램을 작성해야 하므로 E를 눌러 편집모드로 들어간다.

그리고 예제 1을 입력해 보기 바란다.

〔예제 1〕

```
1 ' --- SAMPLE 1 ----------------
2 PRINT "*"
3 PRINT "**"
4 PRINT "***"
5 PRINT "****"
6 PRINT "*****"
7 END
```

위의 행번호는 프로그램을 설명하기 위해 필자가 붙여둔 것이므로 입력하지 않아도 된다.

이 프로그램을 설명하면 다음과 같다.

- 1 : '(키 옆에 있는 작은 따옴표이다)는 주석문을 나타내는 기호로 '이 문장은 프로그램과는 무관하며, 알기 쉽게 설명을 붙여둔 것이므로 실행하지 말아라'는 의미를 갖는다. 베이직에서는 이 작은 따옴표가 붙은 문장은 실행하지 않는다. 그러므로 프로그램의 중간 중간에 자유로이 주석을 붙여둘 수 있다.
- 2-6 :PRINT문은 그 다음의 '겹따옴표로 묶은 부분을 그대로 화면에 출력하라'는 의미이다. 이때 겹따옴표는 화면에 나타나지 않고 그 안의 부분만 나타나게 되며 베이직은 화면에 출력한 후에는 자동적으로 줄을 바꾸게 된다. 겹따옴표 자체를 화면에 나타내는 방법은 다음에 배우게 된다.
- 7:END문은 '프로그램의 실행을 끝내라'라는 의미의 명령어로 END문이 맨 나중에 올 경우에는 생략해도 된다.

이것이 프로그램이다. 이 프로그램을 실행시키면 다음과 같은 결과가 화면에 출력될 것이다.

```
*
**
***
****
*****
```

그러면 실제로 위의 프로그램을 실행하여 이런 결과가 나타나는지 확인을 해보자. 프로그램을 실행하기 전에 먼저 이 프로그램을 디스크에 저장한다. 먼저 〈F2〉키를 누른다. 그러면 다음과 같은 메시지가 나오게 된다.

```
Rename NONAME
A:NONAME
```

이 프로그램의 이름을 지정하지 않아 NONAME.BAS로 되어 있는데 새로운 이름을 입력하겠느냐는 의미이다. 우리는 이름을 지정할 것이므로 바로 이 상태에서 (커서를 이동하지 않는다) B : SAMPLE-1이라고 입력한다. 그러면 우리가 첫 문자를 입력하자마자 A : NONAME은 사라지고 그자리에 우리가 입력한 문자들이 나타나게 된다.

마지막으로 ↵키를 입력하면 B : 드라이브에 불이 오고 드라이브가 돌아가는 소리가 들릴 것이다. 터보 베이직은 우리가 입력한 프로그램을 B : 디스켓에 SAMPLE-1.BAS라는 이름으로 저장하게 된다.

프로그램을 디스크에 저장했으면 이제는 실행을 할 차례이다. 〈Alt-R〉을 누른다. 그러면 화면이 바뀌고 실행 결과가 나타나게 된다. 이때 아무키나 누르면 다시 에디터로 돌아온다. 이 에디터 모드에서 〈F6〉키를 누르면 조금전에 실행했던 화면을 다시 볼 수 있다.

프로그램은 실행하기 전에 반드시 디스크에 저장해두는 것이 안전하다. 왜냐하면 때로는 프로그램의 수행 중 폭주를 하는 경우가 있고, 이 때는 컴퓨터를 껐다가 다시 켜야하므로 애써 입력한 프로그램을 모두 잃어버릴 수가 있기 때문이다.

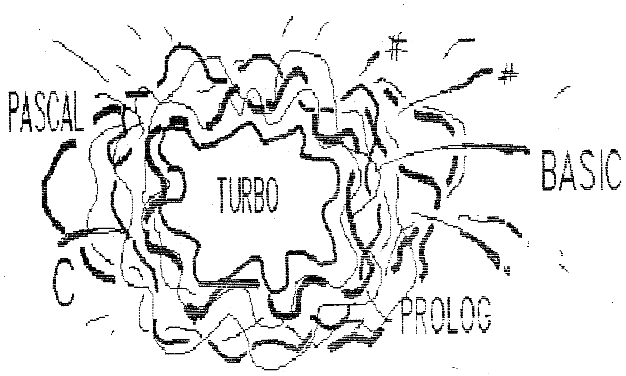
프로그램의 로드

〈Alt-F〉를 눌러서 화일 메뉴로 간다. 거기에서 New를 선택하라. 그러면 지금까지 입력한 프로그램이 모두 지워진다. 여기에서 Load 옵션을 선택하라. 그러면 화일 이름을 물어 올 것이다.

이때 다음과 같이 입력한다.

```
Load File Name : B:SAMPLE-1
```

그러면 B : 드라이브에서 우리가 작성해 둔 프로그램을 불러들이게 된다. 이제 프로그램의 작성 및 실행방법에 관한 전반적인 환경을 알게 되었다. 다음호부터는 이를 기반으로 실제 터보 베이직의 프로그래밍을 익히기로 하자.



전화번호부 프로그램(Ⅲ)

박성현/디베이스 프로그래머

이번호에서는 전화번호부 프로그램을 완결하여 완전한 전화번호부를 이용할 수 있게 하고자 한다. 이번호에 소개될 프로그램은 입력과 스크린 출력 및 프린트 출력에 관한 것인데 전화번호부 입력의 형식을 보여주는 INPUT.prg는 **리스트** 1에, 입력된 변수를 화면에 보여주는 SCREEN. prg는 **리스트** 2에, 입력된 전화번호를 프린트에 출력하도록 하는 OUTPUT.prg는 **리스트** 3에, 그리고 출력할 때 우편에 이용할 수 있는 라벨의 출력과 우편발송에 쓰일 수 있는 봉투를 만들수 있는 프로그램인 OUTPUT 2.prg와 OUTPUT3 prg를 **리스트** 4와 5에 나타내었다.

각 프로그램의 분석은 행번호를 참고로 한 것이며, 모든 프로그램이 완벽하지 않듯이 이 프로그램을 이용하여 독자 나름대로 개성에 맞고 유용한 프로그램을 작성해 보기 바란다.

INPUT.prg 의 분석

(7) ans =[G]

ans라는 변수가 'G'일 때까지 계속하기 위한 ans 변수의 초기값을 설정한다.

(8–52) DO WHILE ans =[G] -- ENDDO

계속 입력을 위한 LOOP. 40행에서 'G'를 입력하게 되면 LOOP의 순환이 된다.

(9–15) mname =SPACE(10) …

필드에 해당되는 형태로 변수의 초기값을 지정해 놓는다. KEY IN과 동시에 데이터베이스 화일에 입력시키는 것

〔리스트 1〕 INPUT.prg

```
 1 * Program..: INPUT.prg
 2 * Author...: Bahk Seonghyeon.
 3 * Date.....: 1989. 4. 5.
 4 * Notes....: 전화 번호부 입력 프로그램
 5 * Notice...: Called from MAIN.prg
 6
 7 ans=[G]
 8 DO WHILE ans=[G]
 9 ¦    mname  = SPACE(10)
10 ¦    mcomp  = SPACE(40)
11 ¦    mjik   = SPACE(10)
12 ¦    maddr  = SPACE(40)
13 ¦    mwoo   = [   -   ]
14 ¦    mtel1  = SPACE(15)
15 ¦    mtel2  = SPACE(15)
16 ¦
17 ¦    SET COLOR TO W/N+
18 ¦    CLEAR
19 ¦    @  3,8  SAY REPLICATE([-],64)
20 ¦    @  4,22 SAY "전   화      번   호   부      입  력"
21 ¦    @  5,8  SAY REPLICATE([-],64)
22 ¦    @ 19,8  SAY REPLICATE([-],64)
23 ¦    @ 21,8  SAY REPLICATE([-],64)
24 ¦    ans=[C]
25 ¦    DO WHILE ANS=[C]
26 ¦    ¦    SET COLOR TO W/N,W/N+
27 ¦    ¦    *---------------*
28 ¦    ¦    DO SCREEN  with 7
29 ¦    ¦    *---------------*
30 ¦    ¦    READ
31 ¦    ¦    @ 20,10 SAY "계속기록(G)      기록(Y)      수정(C)      취소(N)"
32 ¦    ¦    SET COLO TO W/N+
33 ¦    ¦    @ 20,19 SAY [G]
34 ¦    ¦    @ 20,33 SAY [Y]
35 ¦    ¦    @ 20,47 SAY [C]
36 ¦    ¦    @ 20,61 SAY [N]
37 ¦    ¦    ans=[?]
38 ¦    ¦    DO WHILE .NOT. ans$[GYNC]
39 ¦    ¦    ¦    ANS=[?]
```

```
40 ¦    ¦    ¦    @ 20,68 GET ans PICT [!]
41 ¦    ¦    ¦    READ
42 ¦    ¦    ENDDO
43 ¦    ¦    @ 20,10 SAY SPACE(65)
44 ¦    ENDDO
45 ¦
46 ¦    IF ANS$[GY]
47 ¦    ¦    APPEND BLANK
48 ¦    ¦    REPLACE name with mname, comp with mcomp , jik   with mjik
49 ¦    ¦    REPLACE addr with maddr, woo  with mwoo
50 ¦    ¦    REPLACE tel1 with mtel1, tel2 with mtel2
51 ¦    ENDIF
52 ENDDO
53 * End-of-file INPUT.prg
```

〔리스트 2〕 SCREEN.prg

```
1 * Program..: SCREEN.PRG
2 * Author...: Bahk Seonghyeon.
3 * Date.....: 1989. 4. 5.
4 * Notes....: 전화 번호부 입력,열람,수정 양식
5 * Notice...: Called from INPUT , READ
6
7 PARAMETER  pline
8 @ pline+0 ,10 SAY "성          명  : " GET mNAME
9 @ pline+2 ,10 SAY "근   무   처  : " GET mCOMP
10 @ pline+4 ,10 SAY "직          위  : " GET mJIK
11 @ pline+6 ,10 SAY "주          소  : " GET mADDR
12 @ pline+8 ,10 SAY "우 편 번 호  : " GET mWOO
13 @ pline+10,10 SAY "전 화 번 호  : " GET mTEL1
14 @ pline+10,51                        GET mTEL2
15 * End-of-file SCREEN.prg
```

〔리스트 3〕 OUTPUT.prg

```
5 * Notes....: 전화 번호부 출력 프로그램
6 * Notice...: Called from MENU.prg
7 SET COLOR TO W/N+,W/N+
8 CLEAR
9 @  1,5  TO 22,75 DOUBLE
10 @  2,22 SAY "전    화      번   호   부       출   력"
11 @  3,6  TO  3,74
12 @ 17,6  TO 17,74
13
14 ord=[?]
15 @ 7,21 SAY "명 칭 순 ( 1 )      회 사 순 ( 2 )   " GET ord PICTURE [9]
16 READ
```

이 아니라 미리 변수에 입력시킨 후 확인을 거쳐 데이터베이스 화일에 입력시키기 위함이다.

(9) SPACE(n) 함수

n바이트 만큼의 스페이스로 채워진 캐릭터 스트링을 만든다.

(17) SET COLOR TO W/N+

스크린의 색상을 지정하는 명령이다. SET COLOR TO라고만 하면 디폴트인 W/N, N/W가 된다. 여기서 W/N+, N/W라는 것은 white 글씨에 black 바탕을 표시하며 밝은색으로, get명령과 같이 반대색으로 나오는 부분에 있어서는 black 글씨에 white 바탕색으로 지정한다.

밝은색으로의 지정은 +이며, *를 사용하면 blink(깜박 깜박)하는 글씨로 지정된다. 모노 그래픽 카드를 사용할 때 'SET COLOR TO U'라는 명령으로 밑줄을 나타내고 컬러 그래픽 카드를 사용할 때는 BLUE 명령을 사용하여 밑줄을 나타낸다. 각 컬러에 해당되는 코드는 매뉴얼을 참고하기 바란다.

(18) CLEAR

입력 화면을 표시하기 전에 스크린 전체를 지운다.

(19-23) @3,8 SAY REPLICATE([-],64)…

지정한 위치의 스크린에 프린트한다. [-]자를 64개로 나타내게 한다.

(19) REPLICATE(〈expC〉, n)

〈expC〉의 스트링을 n번 만큼 반복하는 스트링으로 바꾸는 함수이다. 원래의 〈expC〉는 1바이트의 문자만이 아니라 여러문자라도 상관이 없으므로 한글의 반복도 가능하다. 특히 그래픽 코드를 사용하여 bar graph를 그릴때 유용하게 사용된다.

(24) ans=[C]

51행에서 'C'를 선택할 때 반복시키기 위한 초기값 설정

(25-44) DO WHILE ANS=[C] …ENDDO

51행에서 'C'를 선택할 때 반복시키기 위한 LOOP.

(26) SET COLOR TO W/N, W/N+

GET/READ 명령에 의해 입력되는

```
IF .NOT. ord$[12]
¦    RETURN
ENDIF
SET COLOR TO W/N+*
@ 9,29 SAY "프 린 트 중 입니다."
SET ORDER TO &ord
pno=1
SET MARGIN TO 3

SET DEVICE TO PRINTER
@ 1,0 SAY CHR(27)+"h"+CHR(27)+"W"
DO WHILE .NOT. EOF()
¦
¦    @ 4,30 SAY "-------------------------------------"
¦    @ 5,30 SAY CHR(27)+"D"+"전  화    번  호  부"
¦    @ 6,30 SAY "-------------------------------------"
¦    @ 7,80 SAY 'PAGE NO. : '+ STR(pno,2)
¦    *                    1         2         3         4         5         6         7         8         9
¦    *           012345678901234567890123456789012345678901234567898012345678901234567890123456789012345678901
¦    *            *---10---*  *---10---*   *--------------------40--------------*       *------15-----*
¦    @ 8,0  SAY  "=========================================================================================="
¦    @ 9,0  SAY  "성  명      직  위      근      무      처                                  전화번호1 "
¦    @ 10,0  SAY  "            우편번호     주                소                                 전화번호2 "
¦    @ 11,0  SAY  "------------------------------------------------------------------------------------------"
¦
¦    ln=13
¦    DO WHILE ln<58 .OR. .NOT. EOF()
¦    ¦    col1=15-LEN(TRIM(tel1))
¦    ¦    col2=15-LEN(TRIM(tel2))
¦    ¦
¦    ¦    @ ln,1       SAY CHR(27)+"B"+name+CHR(27)+"b"
¦    ¦    @ ln,14      SAY jik
¦    ¦    @ ln,28      SAY comp
¦    ¦    @ ln,77+col1 SAY TRIM(tel1)
¦    ¦    ln=ln+1
¦    ¦    @ ln,14      SAY woo
¦    ¦    @ ln,28      SAY addr
¦    ¦    @ ln,77+col2 SAY TRIM(tel2)
¦    ¦    ln=ln+2
¦    ¦    SKIP
¦    ENDDO
¦
¦    @ ln,0  SAY "=========================================================================================="
¦    pno=pno+1
ENDDO

EJECT
SET DEVICE TO SCREEN
RETURN
* End-of-file OUTPUT.prg
```

〔리스트 4〕 OUTPUT2.prg

```
* Program..: OUTPUT2.prg
* Author...: Bahk Seonghyeon.
* Date.....: 1989. 4. 5.
* Notes....: 우편발송 라벨 출력 프로그램
* Notice...: Called from MENU.prg

SET COLOR TO W/N+,W/N+
CLEAR
@  1,5      TO 22,75 DOUBLE
@  3,6      TO  3,74
@ 17,6      TO 17,74

ord=[?]
@ 7,21 SAY "명 칭 순 ( 1 )      회 사 순 ( 2 )   " GET ord PICTURE [9]
READ

IF .NOT. ord$[12]
¦    RETURN
ENDIF
SET COLOR TO W/N+*
@ 9,29 SAY "프 린 트 중 입니다."
SET ORDER TO &ord
SET MARGIN TO 3

SET DEVICE TO PRINTER
@ 1,0 SAY CHR(27)+"h"+CHR(27)+"W"
DO WHILE .NOT. EOF()
¦
¦    ln=3
¦    DO WHILE ln<58 .OR. .NOT. EOF()
¦    ¦    maddr1=addr
¦    ¦    mcomp1=comp
¦    ¦    mname1=name
¦    ¦    mjik1 =jik
¦    ¦    mwoo1 =woo
¦    ¦    skip
¦    ¦    maddr2=addr
¦    ¦    mcomp2=comp
¦    ¦    mname2=trim(name)
¦    ¦    mjik2 =trim(jik)
¦    ¦    mwoo2 =woo
¦    ¦
¦    ¦    if .not. eof()
¦    ¦    ¦    skip
¦    ¦    endif
¦    ¦
¦    ¦    @ ln,8    SAY maddr1
¦    ¦    @ ln,58   SAY maddr2
¦    ¦    ln=ln+2
¦    ¦    @ ln,8    SAY mcomp1
¦    ¦    @ ln,58   SAY mcomp2
```

부분이 밝게 보이게 하기 위해 뒤쪽에 +기호를 넣는다.

(28) DO SCREEN WITH 7

SCREEN.prg를 실행시키는 것으로 파라미터 값을 7로 준다. SCREEN.prg는 프로그램을 부르면서 파라미터 값을 주어야만 하도록 되어 있으므로 반드시 파라미터 값을 함께주어야 한다. 위의 SCREEN.prg는 한번만 사용하는 것이 아니고 조회, 수정할 때 두번 더 사용하기 때문에 따로 화일을 떼어내 사용하면 프로그램 스펙을 줄일 수 있게 된다.

여기서 이 부분을 .FMT로 하여 SET FORMAT TO SCREEN으로 하여도 명령방식이 거의 유사하나 라인, BOX를 .FMT 화일에 다시 그려 주어야 하므로 FMT를 사용하는 것보다 위와 같이 프로그램 화일로 하는 것이 편리한 경우가 많다.

(30) READ

SCREEN. prg에 의해 디스플레이된 것을 입력받게 한다.

(31) @20,10 SAY "계속기록(G) 기록(Y) 수정(C) 취소(N)"

선택 메뉴를 보여준다.

(32) SET COLOR TO W/N+

31행의 메뉴를 선택해야 하는 부분을 밝게 표시하기 위해 COLOR를 지정한다.

(33-36) @20,19 SAY [G] …

선택하여 입력할 키만 밝게 보이게 하기 위해 밝은 글씨로 지정한 후 다시 디스플레이 한다.

(37) ans=[?]

메뉴 선택을 위한 초기값 지정(G, Y, C, N이 아닌 문자이면 된다)

(38-43) DO WHILE. NOT. ans$[GYNC] … ENDDO

입력이 잘못된 경우를 위한 LOOP ans가 'G, Y, C, N'이 아닌 동안 계속 수행한다.

(39) ANS=[?]

입력이 잘못되었을 때 입력한 내용 대신 '?'를 보여주기 위해 변수값을

```
52 |     |     ln=ln+2
53 |     |     @ ln,8     SAY CHR(27)+[D]+mname1+CHR(27)+[d]+[ ]+mjik1+[ 님 귀하]
54 |     |     @ ln,58    SAY CHR(27)+[D]+mname2+CHR(27)+[d]+[ ]+mjik2+[ 님 귀하]
55 |     |     ln=ln+3
56 |     |     @ ln,18    SAY [(우)]+mwoo1
57 |     |     @ ln,68    SAY [(우)]+mwoo2
58 |     |     ln=ln+5
59 |     |
60 |     ENDDO
61 |
62 ENDDO
63
64 EJECT
65 SET DEVICE TO SCREEN
66 RETURN
67 * End-of-file OUTPUT.prg
```

〔리스트 5〕 OUTPUT3.prg

```
 1 * Program..: OUTPUT3.prg
 2 * Author...: Bahk Seonghyeon.
 3 * Date.....: 1989. 4. 5.
 4 * Notes....: 우편발송 봉부 출력 프로그램
 5 * Notice...: Called from MENU.prg
 6
 7 SET COLOR TO W/N+,W/N+
 8 CLEAR
 9 @  1,5      TO 22,75 DOUBLE
10 @  2,22 SAY "우  편   발  송    봉  부     출  력"
11 @  3,6      TO  3,74
12 @ 17,6      TO 17,74
13
14 ord=[?]
15 @ 7,21 SAY "명 칭 순 ( 1 )      회 사 순 ( 2 )   " GET ord PICTURE [9]
16 READ
17
18 IF .NOT. ord$[12]
19 |    RETURN
20 ENDIF
21 SET COLOR TO W/N+*
22 @ 9,29 SAY "프 린 트 중 입니다."
23 SET ORDER TO &ord
24 SET MARGIN TO 3
25
26 SET DEVICE TO PRINTER
27 @ 1,0 SAY CHR(27)+"h"+CHR(27)+"W"
28 DO WHILE .NOT. EOF()
29 |    @  4,5  SAY CHR(27)+[D]+[퍼 스 널 컴 퓨 터]
30 |    @  6,5  SAY [서울 특별시 용산 갈월동 17 번지(세기빌딩)
```

재지정

(40) @20,68 GET ans PICT [!]

ans 값을 대문자로 바꿔서 입력을 받게 한다.

(43) @20,10 SAY SPACE(65)

선택 메뉴 부분만 지운다.

(46-51) IF ANS$[GY] …ENDIF

기록에 해당하는 명령을 하였을 경우 수행한다.

(47) APPEND BLANK

현재 사용하고 있는 데이터베이스 화일에 빈공간으로 된 레코드 하나를 추가한다. 단독으로 APPEND 명령을 사용하면 디베이스의 입력모드가 나오면서 레코드를 추가시키지만 APPEND BLANK라고 명령하면 현재 사용중인 데이터베이스 화일의 끝부분을 BLANK로 채운채 레코드 하나를 추가시키게 된다.

(48-50) REPLACE name with mname…

방금 새로 추가한 레코드를 각 필드에 해당하는 변수로 바꾸어 주는 명령이다. 이때는 인덱스 화일도 함께 열려있으므로 이 순간에 인덱스의 구조도 다시 정렬된다.

여기서 ';'는 문장이 길 경우 계속해서 다음 라인에 표시하기 위해 다음의 RETURN 값을 무시하라는 기호이다. (디베이스는 RETURN에 의해 한 명령이 끝남을 표시한다.)

SCREEN.prg의 분석

(7) PARAMETER pline

프로그램을 호출할 때 반드시 pline이라는 변수를 함께 지정하게 하고, 반드시 파라미터 값을 받게 한다. 이 명령을 사용할 때에는 항상 프로그램의 맨위에 넣어야 한다.

(8-14) @pline +0, 10 SAY "성명" ; GET mNAME…

파라미터로 보내준 변수를 라인으로 하여 화면에 지정한 변수를 디스플레이하여 입력받을 준비를 한다.

```
31 ¦    @  8,5  SAY [Tel  (02)754-1044(대)     Fax  (02)757-1048]
32 ¦    @ 10,5  SAY [은행지로 : 300667          대체구좌 : 526772]
33 ¦    @ 11,30 SAY [140-150]
34 ¦
35 ¦    @ 40,58 SAY maddr2
36 ¦    @ 42,58 SAY mcomp2
37 ¦    @ 44,58 SAY CHR(27)+[D]+trim(name)+CHR(27)+[d]+[ ]+trim(jik)+[ 님 귀하]
38 ¦    @ 47,78 SAY [(우)]+mwoo2
39 ¦
40 ¦
41 ENDDO
42
43 EJECT
44 SET DEVICE TO SCREEN
45 RETURN
46 * End-of-file OUTPUT3.prg
```

OUTPUT.prg 의 분석

(7) SET COLOR TO W/N+, W/N+

GET에 의해 반전되는 색이 검은 배경에 밝은 흰글씨로 나오게 하기 위해 W/N+를 사용한다. SET INTENSITY OFF로 지정해도 같은 효과가 된다.

(8) CLEAR

열람 화면이 나오기전 화면을 깨끗이 지운다.

(9) @1,5 TO 22,75 DOUBLE

모니터의 경우 텍스트 모드에서 80행 25라인의 CELL을 가지고 있으며, 좌측 상단의 끝점이 0,0이고 우측 하단의 끝부분은 24,79가 된다. 사각형을 그리는데는 좌측 상단의 끝점과 우측 하단의 끝점 2점만 있으면 사각형을 그릴 수 있으므로 위의 1,5의 좌측 상단 점과 22,75의 우측 하단 점을 연결하여 사각형의 BOX를 그린다. DOUBLE이라는 옵션을 주면 2줄짜리 BOX를 그린다.

8비트 조합형, 완성형 한글에서는 한글코드가 이 세미그래픽을 포함하고 있으므로 깨진 한글이나 한자가 나오므로 이 부분은 다음과 같이 수정한다.

```
@  1,5  SAY REPLICATE('=',70)
@ 22,5  SAY REPLICATE('=',70)
@  2,5  SAY '¦'
@  2,75 SAY '¦'
@  3,5  SAY '¦'
  ............
@ 21,5  SAY '¦'
@ 21,75 SAY '¦'
```

물론 2바이트 그래픽 코드를 찾아 그려도 된다. 또 한가지 방법은 TEXT…ENDTEXT를 사용하는 것으로 TEXT 다음의 라인부터 ENDTEXT 전 라인의 내용을 디스플레이 해주는 명령이다. 그러므로 워드 프로세서로 화면 전체를 디자인해서 사용하면 된다.

다음을 테스트해 보라

```
TEXT
 ------------------------------------------------
|           ======================               |
|--▤▤▤  전  화  번  호  부  출  력  ▤▤▤--|
|           ======================               |
|                                                |
 ------------------------------------------------
ENDTEXT
```

(11-12) @3,6 TO 3,74

1줄짜리 박스를 그리는 명령이지만 같은 라인을 지정했으므로 가로줄을 그리게 된다. 11행의 BOX안에 들어가는 가로 줄이므로 컬럼을 하나씩 안쪽으로 정해야 한다.

(14) ord=[?]

변수 ord의 값을 '?'로 초기화한다.

(15-16)

ord 변수를 새로 입력받는다. PICTURE[9]는 숫자만 입력받고자 할 때 사용한다. READ는 지금까지 나온 GET을 입력할 때까지 기다리고 그 내용을 받아들인다.

(18-20) IF.NOT.ord$[12] …ENDIF

ord 변수값이 '1' 또는 '2'가 아닐 경우는 메인 프로그램으로 복귀하게 한다.

**(21) SET COLOR TO W/N+*

검은 배경에 흰글씨로 나오면서 점멸되게 한다. '*'은 blink를 위한 표시이다.

(22) @9,29 SAY "프린트중입니다"

프린트 중임을 알리는 메시지를 보여준다.

(23) SET ORDER TO &ord

열려있는 인덱스 화일중 n번째 화일을 마스터 인덱스 화일로 지정한다. n이 2라고 하면 위의 인덱스 화일 중 CCOMP.ndx가 마스터 인덱스 화일이 되어 그순간부터 CARD.dbf의 배열은 회사순으로 바뀌게 된다.

다른 배열의 인덱스 화일로 바꾸고자 할 때 이와같이 인덱스 화일을 새로 열지 않고 한꺼번에 같이 열어 놓은 후 마스터 인덱스 화일만 바꾸어 주면 자주 화일을 열고 닫는 시간을 줄이게 된다. 또 화일을 같이 열어 주어야 데이터가 바뀌거나 추가될 때 인덱스 화일도 함께 재배열 된다. &ord는 ord 라는 변수에 들어 있는 내용을 명령으로 인식하게 해준다.

&(macro)의 기능은 변수의 내용이 문자인 경우에 기억변수의 내용을 치환하는데 사용한다.

(24) pno=1

페이지 번호를 초기화한다.

(25) SET MARGIN TO 3

프린트의 왼쪽 여백을 숫자만큼 조정한다. 화면에는 영향을 미치지 않고 프린터에만 영향을 준다. 프린터의 0, 3이 이후로는 0, 0으로 인식되게 한다.

(27) SET DEVICE TO PRINTER

@…SAY 명령을 프린터에 출력을 한다. row, column은 프린터의 위치를 지정한다. 프린터에서는 뒤로 움직일 수 없으므로 row의 숫자가 작아지면 다음 페이지에 프린트가 되므로 row위치를 잘 지정해야 한다.

column의 숫자는 작은 숫자가 나중에 나와도 되지만 속도를 빠르게 하려면 작은 숫자가 먼저 나와야 한다.

(28) @1,0 SAY CHR(27)+"h"+CHR(27)+"W"

엡슨 프린터에 프린터 제어 코드를 주는 것으로(TG 모드인 경우에만 해당함) 실제로 프린트되는 것이 아니라 프린터에 주는 명령이다. CHR(27)은 Esc-코드이므로 Esc+"h"는 한글 모드로 세팅하는 것이며 Esc+"W"는한글과 영문의 비율을 1:2로 세팅해 주는 것이다.

프린터의 모드를 딥 스위치로 지정해 놓았더라도 다른 프로그램을 프린트한 후는 모드가 바뀔수 있으므로 반드시 이 제어코드를 포함시켜 사용하는 것이 안전하다.

베이직에서의 CHR$(27);"h";CHR(27);"W"와 같은 명령이다. 엡슨 KS-SM 모드 또는 다른 프린터의 경우 매뉴얼을 참고하여 프린터 제어코드를 사용하기 바란다.

(29-62) DO WHILE. NOT. EOF( )…ENDDO

CARD. dbf의 위치가 end-of-file 이 아닌 동안 계속

(32) @5,30 SAY CHR(27)+"D"+"전화번호부"

CHR(27)+"D"는 가로로 2배 확대를 위한 프린터 제어코드이다. CHR(27)+"d"는 가로로 2배 확대를 취소하여야 하지만 이 코드는 라인이 바뀜에 따라 그 기능이 상실되므로 이 명령은 취소할 수 있다.

(34) @7,80 SAY 'PAGE NO.:'+STR(pno, 2)

페이지 번호를 프린트한다. STR( )은 숫자를 문자로 바꿔주는 함수로 뒤의 2는 2자리 문자 변수로 바꿔주도록 하는 것이다.

(35-37) *

프린터에 각 필드가 위치할 컬럼을 정할 때 헤드부분에 이와같이 컬럼 위치와 필드의 길이를 그려놓으면 컬럼 위치를 정하는데 편리하다.

(31-41)

HEAD 부분을 프린트한다.

(43) ln=13

프린트 시작할 라인을 미리 지정해 놓는다.

(44-58) DO WHILE ln <58 .OR. .NOT. EOF( )…ENDDO

라인번호가 58보다 작거나 end-of-file이 아닌 동안 계속된다. 프린터의 행수는 디폴트로 66라인으로 되어 있으므로 0-65까지 프린터하고 다음은 새로운 페이지가 된다. 상하 여백을두기 위해 58보다 크거나 같으면 LOOP에서 빠져 나와 다시 헤드 부분을 프린트하게 한다. 이때 헤드부분을 프린트할 때는 31행에 의해 라인번호 4에서 프린트하므로 새로운 페이지에 프린트하게 된다.

(45-46)COl1=15-LEN(TRIM[tel1])

(51), (55) 행에서 전화번호의 뒤 공백을 없애고 뒤의 오른쪽 끝부분에 맞추기 위해 공백의 길이를 계산한다. 계산하는 방법은 본래의 크기 15에서 공백을 제거한 데이터의 차이를 구한다.

(48) :@ln,1 SAY CHR(27)+"B"+name+CHR(27)+"b"

성명을 프린트한다. CHR(27)+"B"는 굵은 글씨의 시작코드이고, CHR(27)+"b"는 굵은 글씨의 해제코드이다.

프린터에서는 2번 인자를 하게 된다. 이것은 엡슨 계열의 프린터에서 해당되는 것이니 다른 프린터는 해당 프린터의 매뉴얼을 참고하여 입력해야 한다.

(49) @ln,14 SAY jik

직위를 프린트한다.

(50) @ln,28 SAY comp

회사명을 프린트한다.

(51): @ln,77+col1 SAY TRIM(tell)

전화번호가 제거된 공백수만큼 오른쪽으로 밀려서 프린트 된다. TRIM( )은 문자의 뒤 여백을 제거해 주는 함수이다.

```
56 |   |   ln=ln+2
57 |   |   SKIP
58 |   ENDDO
```

(52) ln=ln+1

라인번호를 증가한다. 한 레코드를 2줄에 걸쳐 프린트하게 되므로 다음칸에 프린트하기 위해 1을 더해 준다.

(53) @ln,14 SAY Woo

우편번호를 프린트 한다.

(54) @ln,28 SAY addr

주소를 프린트한다.

(55) @ln,77+co12 SAY TRIM(te12)

전화번호의 뒤여백 만큼 오른쪽으로 당겨서 프린트한다. 오른쪽 끝에 맞추기 위한 것이다.

(56) ln=ln+2

한 레코드의 내용을 프린트한 후 1칸을 띄우기 위해 2를 더해준다.

(57) SKIP

다음 레코드를 읽기 위해 다음 레코드로 내려간다. 마스터 인덱스의 순서대로 차례로 진행된다.

(60) @ln,0 SAY "…………………"

한 페이지가 끝났으므로 줄을 긋는다.

(61) pno=pno+1

페이지 번호를 하나 증가시킨다.

(64) EJECT

프린터가 다음 페이지 맨 위에 오도록 종이를 이동시킨다.

(65) SET DEVICE TO SCREEN

@… SAY 명령이 모니터에 출력되게 한다.

(66) RETURN

메인 프로그램으로 복귀한다.

OUTPUT2.prg 의 분석

여기까지 프로그램을 해보았다면 이제는 출력 프로그램은 무슨 형태이든 만들 수 있을 것이다. 디베이스에서 제공되는 report form, label form을 사용하는 것보다 자유자재로 출력형태를 만들어 낼 수 있다.

그럼 간단히 프린트 용지에 라벨을 프린트해서 봉투에 붙이는 프로그램을 만들어 보자. 지난호에서의 메인 메뉴는 이 프로그램이 연결되어 있지 않으므로 메인 메뉴를 적당히 수정하여 실행해 보기 바란다. 라벨은 프린트 용지보다 작으므로 한번에 두개의 라벨을 프린트해야 한다. OUTPUT.prg를 이용하여 필요한 부분만 고치면 간단히 만들어 낼 수 있을 것이다.

(31–35)

한번에 2개의 레코드를 프린트하기 위해 첫번째 필드의 내용을 변수로 만들어 기억한다.

(33–34, 39–40)

이름과 직위는 뒤의 공백을 없애주기 위하여 trim(C) 함수를 사용하여 변수로 지정한다.

(36) skip

레코드 하나를 이동시켜 두번째의 것을 변수로 기억하기 위하여 준비한다.

(37–41)

한번에 2개의 레코드를 프린트하기 위해 두번째 필드의 내용을 변수로 만들어 기억한다.

(43–45)

첫번째의 레코드가 마지막 레코드였다면 다음 레코드로 skip하여 이미 end-of-file로 갔기 때문에 또다시 skip 한 명령을 사용한다면 에러가 발생되므로 에러를 방지하기 위하여 end-of-file이 아닌 경우에만 skip 명령을 사용한다.

첫번째의 레코드가 마지막 레코드였고 그다음에 skip하여 end-of-file에 와 있다면 두번째 레코드는 아무 내용도 없는 변수로 저장된다.

(47–58)

2개의 레코드를 동시에 프린트한다.

OUTPUT3.prg의 분석

다음은 우편발송용 봉투의 형태로 된 프린트 용지를 이용해서 직접 봉투를 프린트하는 프로그램이다. 이것은 한 페이지에 하나의 레코드를 프린트하는 것이므로 다른 프로그램보다 더 간단하다는 것을 알 수 있을 것이다. 원하는 모양대로 마음대로 바꾸어 사용할 수 있을 것이다.

(28) DO WHILE. NOT. EOF( )

한 페이지에 하나의 레코드를 프린트하기 때문에 몇번째 라인인가는 기억할 필요가 없다.

(29–33)

보내는 사람의 주소를 프린트한다.

(35–38)

받는 사람의 주소를 프린트한다.

여기까지 공부를 다했다면 원하는 데이터를 원하는 모양으로 입력과 출력을 할 수 있는 프로그램을 간단히 만들어 낼 수 있을 것이다. 컴퓨터에 대한 구조와 다른 랭귀지를 전혀 몰라도 간단한 디베이스에 대한 지식만으로 디베이스 프로그래머가 되었다는 마음으로 자신이 필요한 프로그램은 자신이 직접 작성해서 사용해 보길 바란다.

지금까지 이 강좌를 읽어주신 독자 여러분께 감사드리며 이렇게 간단한 프로그램은 디베이스를 이용해서 프로그램을 할 수 있지만 이러한 디베이스 자체는 C언어로 만들어졌다는 것을 상기하며 이 글을 읽는 독자는 C언어를 익혀서 디베이스보다 훌륭한 프로그램을 만들어 주길 당부하고 싶다.

디베이스를 사랑하는 필자가 디베이스를 사랑하지 않게 되길 바라면서…

(Hi) PC

□ 알림 □

지금까지 8회에 걸쳐 게재되었던 본 연재는 이 번호를 마지막으로 끝마치게 되었습니다.

그동안 좋은 원고를 집필해 주신 박성현님께 감사드리며 다음호부터는 본 난을 좀더 새로운 내용으로 꾸며 독자 여러분을 찾아 뵐 것을 약속드립니다.

저수준 화일 입출력 함수

이용일 / 한국전자통신연구소

C언어는 함수적인 언어(Functional Language)라고도 한다. 일반적으로 다른 프로그래밍 언어는 명령문을 얼마나 잘 활용하느냐에 따라 최적의 프로그램을 작성할 수 있지만, C언어에서는 수백개에 달하는 함수가 라이브러리로서 지원되고 있어, 다른 프로그래밍 언어와는 달리 함수의 이용법을 확실히 숙지하여 라이브러리를 최대로 활용하는 것이 C언어에 의한 최적의 프로그램을 작성할 수 있는 지름길인 것이다.

따라서 C언어에 의해 간단한 프로그램을 작성하려는 사용자에게 필요한 것은, 각 함수의 정의 및 형식 그리고 그 활용 예이다. 그러나 조금 더 원리적인 것을 공부하여 다양한 기능을 갖춘 프로그램을 작성하려는 사용자에게는, 각 처리계에서 함수를 구체적으로 어떻게 구현하는가 하는 것이 필요할 것이다.

이번 호부터는 본 강좌의 연재를 크게 2가지로 분류하여 설명하고자 한다. 제1부는 기초적인 사용자를 위한 각 함수의 정의 및 형식 그리고 활용예에 관한 설명을 주로 하고, 제2부는 구체적으로 MS-DOS에서 각 함수를 어떻게 실현하고 있는가를 설명할 것이다.

우선 본고에서는 1부에 해당되는 것을 설명하고 2부에 해당하는 것은 다음호에서 설명하기로 한다.

함수의 활용

C에서의 화일조작은, 저수준 화일 입출력 함수와 표준 화일 입출력 함수라는 2종류의 라이브러리 루틴에 의해 지원되고 있다. 이중에서 이번에 설명하는 저수준 입출력 함수는 화일핸들 및 화일속성, 액세스 모드 등을 직접 취급하는 것으로, 다음 연재에서 설명할 표준 화일 입출력 함수의 요청으로 데이터를 바이트단위, 블록단위로 취급하는 경우에 사용된다.

마이크로소프트 C에서 지원하는 저수준 화일 입출력 함수에는 표 1과 같이 많은 것이 준비되어 있다. 여기서는, 이들 중에서 화일조작을 위한 가장 기본적인 처리를 하는 creat, open, read, write, close, unlink 등의 6개의 함수에 대해 구체적으로 설명하고자 한다.

화일조작 위한 기본사항

(1) 화일핸들

MS-DOS에서는, 각 프로세스마다 20개의 화일을 취급할 수 있도록 설계되어 있으며, 저수준 입출력 함수와 같이 시스템 콜을 사용하여 이들 화일에 액세스하는 경우에는, 이 중에서 어느 화일에 액세스할 것인가를 시스템이 관리하고 있는 번호로 지정하게 된다. 바로 이 번호를 화일핸들이라 부르며 0이상의 작은 정수를 사용하여 식별하고 있다. 이렇게 하여 MS-DOS는 트랙 및 섹터와 같은 화일의 실제 할당 등의 관리를 하고, 프로그램 작성자는 이 화일핸들을 사용하여 화일조작을 할 뿐으로 프로그램 작성시의 부담이 매우 가벼워지게 된다.

이 화일핸들을 유용하게 사용하기 위해서는 저수준 입출력 함수 중의 open 함수 등을 이용하여 화일상태 및 관리 정보를 설정해야 하는데, MS-DOS 및 마이크로소프트 C의 저수준 입출력 함수에서는 표 2와 같은 5개의 화일핸들을 미리 설정해 놓고 이들을 사용할 수 있도록 하고 있다.

(2) 화일속성

MS-DOS가 관리하고 있는 화일은, 각각의 화일이 어떤 성질을 갖고 있는 화일인가를 나타내기 위하여 화일속성이라고 불리는 것을 갖고 있다. 그들을 정리하면 표 3과 같으며, 저수준 입출력 함수를 사용하여 화일속성을 설정하려면, 인클루드화일 stat.h에 정의되어 있는 식별자를 사용하여 지정한다.

(3) 액세스모드

화일을 어떻게 사용할 것인가를 나타내는 것을 액세스모드라고 한다. 구체적인 지정방법은 표 4에 보인 9개의 식별자를 C언어의 OR 연산자인 "|"로 조합시켜 지정하여, 저수준 입출력 함수에 그 화일에 대한 액세스방법을 알려준다. 이들 액세스모드는 인클루드 화일 fcntl.h에 정의되어 있다.

① creat

〈사용방법〉
ret=creat(fname, pmode)
여기서,
int ret : 성공하면 화일핸들, 에러이면 −1
char ＊fname : 작성하려는 화일 이름
int pmode : 화일에 붙는 속성

creat 함수는, 인수 fname으로 지정된 화일을, 인수 pmode로 지정된 화일 속성으로 화일을 새로 작성하기 위한 함수이다. 또 지정된 화일이 이미 존재하고 있는 경우에는—그것이 디렉토리가 아니며, 화일속성으로 써넣기를 허가한 경우에 한하여—화일길이를 영(0)으로 하고, 인수 pmode로 지정된 화일 속성을 설정하여 개방(open)한다.

또 creat 함수에 의해 개방된 화일은 읽고 쓰기가 가능한 모드로 되어 있다. creat 함수는 함수처리가 성공했을 때는 개방한 화일의 화일핸들을 돌려주며, 실패했을 경우에는 돌려주는 값(Return Value)으로서 −1을 부여한다. 그리고 creat 함수 뿐만 아니라, 일반적으로 저수준 화일 입출력은 에러가 발생한 경우 돌려주는 값으로서 0 이외의 값(보통은 −1)을 돌려준다.

② open

```
〈사용방법〉
ret=open(fname, mode[, pmode])
여기서,
int ret : 성공하면 화일핸들,
          에러이면 −1
char *fname : 개방하는 화일 이름
int mode : 화일 개방 모드
int pmode : 화일에 붙는 속성
```

open함수는 인수 fname으로 지정된 화일을 인수 mode로 지정된 액세스모드로 화일을 개방한다. 또 open 함수에서는 액세스모드로 O_CREAT를 지정하면 creat 함수와 마찬가지로 화일을 신규 작성할 수 있지만, 이때에 한하여 인수 pmode로 지정한 화일속성이 필요하게 된다.

또 open 함수는 돌려주는 값으로서, 처리에 성공한 경우에는 개방된 화일의 화일핸들을, 그리고 실패한 경우에는 −1을 돌려준다. 인수 mode로 지정한 액세스모드에 대해서는 다음호에서 자세히 설명하겠지만, 여기서 간략하게 설명하겠다. O_RDONLY, O_WRONLY, O_RDWR의 각 액세스모

〔표 1〕 저수준 입출력 함수

분 류	함수의 이름 및 형	기 능
기본적인 조 작	int creat	화일의 신규 작성
	int open	화일을 개방(open)
	int read	화일 읽기
	int write	화일 쓰기
	int close	화일의 클로즈
	int unlink	화일의 삭제
	int rename	화일 이름의 변경
	int setmode	텍스트/바이너리 모드의 설정
Seek 조작	long lseek	화일의 현재 위치 이동
	long tell	화일의 현재 위치 취득
	int eof	화일 끝의 검사
	int chsize	화일 크기의 변경
	long filesize	화일 크기의 취득
리다이렉트 조 작	int dup	화일 디스크립터의 2중화
	int dup2	
화일속성의 조 작	int access	화일 속성의 취득
	int chmod	화일 속성의 변경
	int stat	화일 정보의 취득
	int fstat	화일 정보의 취득
	int umask	화일 속성 마스크의 설정
	int utime	화일 작성일시의 갱신
	int isatty	디바이스 화일 여부 검사
디렉토리 조 작	int chdir	현재 디렉토리의 변경
	char *getcwd	현재 디렉토리의 경로이름 취득
	int mkdir	서브 디렉토리 작성
	int rmdir	서브 디렉토리 삭제

〔표 2〕 표준 입출력과 화일 핸들

표준입력(stdin)	화일핸들 0
표준출력(stdout)	화일핸들 1
표준에러(stderr)	화일핸들 2
표준보조입출력(stdaux)	화일핸들 3
표준프린터(stdprn)	화일핸들 4

〔표 3〕 화일 속성

archive	보존 화일
system	시스템 화일
hidden	숨겨진 화일
readonly	써넣기 금지
directory	서브 디렉토리
volume	볼륨 레이블

〔표 4〕 액세스 모드

0_RDONLY	읽기 만을 수행한다.
0_WRONLY	써넣기 만을 수행한다.
0_RDWR	읽고 쓰기를 한다.
0_APPEND	어펜드 모드로 개방한다.
0_CREAT	지정화일이 없는 경우 작성한다.
0_TRUNC	화일의 크기를 0으로 한 후 처리개시.
0_EXCL	지정화일이 이미 존재하는 경우 에러.
0_BINARY	바이너리 모드로 입출력을 수행한다.
0_TEXT	텍스트 모드로 입출력을 수행한다.

드에 대해서는 항상 이들 중에서 하나만 지정하도록 해야 한다. 또 O_EXCL은 네트워크에서 공유화일의 배타적인 작성을 하는 것이며, 항상 O_CREAT와 동시에 "O_CREAT O_EXCL"과 같이 지정해야 한다. 또 네트워크에서의 공유화일에 대한 상세한 것은 기회를 보아 설명하겠다.

③ read

〈사용방법〉
ret=read(fh, buf, nbytes)
여기서,
int ret : 실제로 읽어들인 바이트 수, 0일 때는 화일 끝, −1은 에러
int fh : 화일핸들
char *buf : 읽어들일 데이터의 버퍼포인터
unsigned nbytes : 읽어들일 바이트 수

read 함수는 인수 fh로 지정된 화일핸들이 가리키는 화일에서, 인수 nbytes로 지정된 바이트 수 만큼 데이터를 읽어들이고, 이것을 인수 buf로 지정된 영역에 넣어두는 함수이다. 또, 그 화일이 텍스트모드로 개방되어 있는 경우에는, 읽어들인 데이터에 대하여 행 끝 및 화일 끝의 변환처리를 수행한 후에 저장한다. 돌려주는 값은 실제로 읽어들인 바이트수이다. 단, 이것은 인수 nbytes로 지정한 바이트 수로 한정되는 것이 아니며, 텍스트모드의 경우에는 변환후의 바이트 수가 돌려진다.

또 읽어들이는 도중에 화일 끝에 도달한 경우에도 nbytes보다 작은 값이 돌려지고, 이후 계속 0이 돌려지지만 이것은 에러가 아니며 인수 buf로 지정된 영역에는 화일끝까지의 데이터가 저장되어 있다. 또 에러가 발생한 경우에는 −1을 돌려준다. 또, 텍스트모드의 경우에 일단 화일 끝에 도달하면(즉, 0x1a를 읽어들인다) 내부의 관리영역에 그것이 기록되기 때문에 콘솔로부터의 입력에서 한번 '^Z'를 입력하게 되면 두번다시 복구할 수 없으므로 주의해야 한다.

④ write

〈사용방법〉
ret=write(fh, buf, nbytes)
여기서,
int ret : 써넣은 바이트 수, 에러이면 −1
int fh : 화일핸들
char *buf : 써넣을 데이터의 버퍼 포인터
unsigned nbytes : 써넣을 바이트 수

write 함수는, 인수 buf에서 지정된 영역의 선두부터 인수 nbytes로 지정된 바이트 수의 데이터를 인수 fh로 지정된 화일핸들이 가리키는 화일에 써넣는 함수이다. 단, 인수 nbytes에 0을 지정한 경우에는 현재의 화일 포인터에서 화일 끝까지의 내용이 잘려지므로 주의가 필요하다.

써넣기에 있어서, 지정된 화일이 텍스트모드인 경우에는 그 데이터에 대하여 행 끝 변환을 한 후에 써넣는다. 돌려주는 값으로서는 실제로 써넣은 바이트 수를 돌리지만 이때 텍스트모드로 변환하기 이전의 바이트 수를 돌려준다. 또 화일핸들이 부적당하다는 등의 에러가 있으면 −1이 돌려진다. write에서는 돌려주는 값에 의한 실제로 써넣은 바이트 수와 인수로 지정한 바이트 수가 일치하지 않는 경우에는 실질적인 에러라고 생각할 수 있다. 또, 그 이유로는 디바이스가 꽉 찼다든지 하는 등의 원인을 생각할 수 있지만, 인수로서 지정한 버퍼의 최후에 MS-DOS의 화일끝 코드(0x1a)가 존재하는 특수한 경우를 생각해 보면 write 함수는 거기까지만 써넣기를 수행하고 돌려주는 값으로서 그 이전에 써넣은 바이트수를 돌려주는 경우도 있다.

⑤ close

〈사용방법〉
ret=close(fh)
여기서,
int ret : 성공하면 0, 에러이면 −1
int fh : 클로즈하는 화일핸들

close 함수는, 인수 fh로 지정된 화일핸들을 해방하여 화일과 화일핸들의 결합을 끊는다. 또 이때, 화일크기 등의 정보를 디렉토리에 써넣는다. 이 처리에 의해 해방된 화일핸들은 다시 이용할 수 있게 되기 때문에 필요가 없어진 화일핸들에 대해서는 그때그때 클로즈(close)를 실행하도록 하는 것이 좋다. close 함수는, 돌려주는 값으로서 성공한 경우에는 0 실패한 경우에는 −1을 돌려준다.

⑥ unlink

〈사용방법〉
ret=unlink(fname)
여기서,
int ret : 성공하면 0, 에러이면 −1
char *fname : 삭제하려는 화일이름

unlink 함수는 인수 fname으로 지정된 화일을 삭제하는 함수이다. 단, 인수 fname으로 지정한 화일은 보통의 화일이어야 하며(디렉토리도 아니고, 디바이스도 아니어야 한다), 또 화일속성으로서 써넣기 허가가 있어야 한다. 그리고 디렉토리를 삭제하는 경우에는 표 1에서 알 수 있듯이 rmdir 함수를 사용한다. 돌려주는 값으로서 성공한 경우에는 0, 실패한 경우에는 −1을 돌려준다.

활용 예

(1) 화일의 바이트와 행 수를 세는 명령어

실제의 예로서 명령어라인에서 지정한 화일의 바이트 수와 행 수를 출력하는 명령어 pcwc(wc는 유닉스의 필터 명령어로서 화일을 읽어들여 그 화일의 바이트 수와 행 수를 출력하는 명령어이다)를 리스트 1에 보인다. 실행예는 화면 1과 같다.

(2) 복수개의 화일을 삭제하는 명령어

MS-DOS의 del명령어는 한개의 화일을 삭제하든가 또는 와일드카드문자(＊, ?)를 사용하여 복수개의 화일을 삭제할 수 있다. 그러나 와일드카드문자를 사용하지 않고는 복수개의 화일을 삭제할 방법이 없다. 그래서 한 명령어라인으로 복수개의 화일을 지정하여 지정된 화일들을 삭제하는 명령어 프로그램을 작성하였다. 프로그램은 리스트 2에 보이고, 화면 2에는 실행 예를 보인다.

〔리스트 1〕 PCWC명령어

```
C>typen pcwc.c   ------typen 명령어는 89년 4월호 참조

/*   pcwc.c ---화일의 바이트 수와 행 수를 세는 명령어     */

#include "stdio.h"
#include "fcntl.h"

main(argc,argv)
int argc:
char **argv;
{
        int fh;
        void wc();

        if (argc == 1) /* 화일 이름을 지정하지 않았다. */
          printf("\n\t usuage : pcwc<화일이름> ...\n");
        else
          while(--argc)
            if ((fh = open(*++argv, O_RDONLY)) == -1)
              printf("\n pcwc:cannot open %s \n", *argv);
            else
            {
            wc(*argv, fh);
            close(fh);
            }
}

void wc(fname,fh)
char *fname:
int fh;
{
        int c;
        long nbytes,lines;

        for (nbytes = lines = OL; (c = _getc(fh)) != EOF; nbytes++)
            if(c == '\n')
                lines++;
        printf("\n\t file names : %s",fname);
        printf("\n\t no. of bytes : %ld",nbytes);
        printf("\n\t no. of lines : %ld",lines);
}

int _getc(fh)
int fh;
{
        unsigned char c;

        return read(fh,&c,1) ? c:EOF;
}
```

〔리스트 2〕 PCdel 명령어

```
C>typen pcdel.c

/*   pcdel.c --한번에 여러개의 화일을 지우는 명령어     */

#include "stdio.h"

main(argc, argv)
int argc;
char **argv;
{
        while (--argc)
          if (unlink(*++argv))
                printf("\n pcdel :cannot delete %s \n", *argv);
}
C>
```

〔화면 1〕 PCWC명령어 실행 예

```
C>pcwc  ---- 인수없이 리턴키를 친다.

      usuage : pcwc <화일이름> ... ---- 사용방법을 알려준다.

C> pcwc pcwc.c ---- pcwc.c 원시프로그램의 크기를 알아보자.

  ** file name : pcwc.c
       no. of bytes : 952
       no. of lines : 47

C>pcwc pcwc.c typen.c ---- 인수를 복수개 지정해보자.
                            여기서, typen.c 프로그램은 '89.4월호 참조
  ** file name : pcwc.c
       no. of bytes : 952
       no. of lines : 47

  ** file name : typen.c
       no. of bytes : 1256
       no. of lines : 50
C>
```

〔화면 2〕 PCdel 명령어 실행 예

```
C>dir/w
 Volume in drive C has no label
 Direcory of C:\

PCWC   C      PCWC   BAK    PCWC   OBJ
PCWC   EXE    PCDEL  C      PCDEL  BAK
PCDEL  OBJ    PCDEL  EXE    TYPEN  EXE
        9 File(s)  1613125 bytes free

C>pcdel pcwc.bak pcwc.obj pcdel.bak pcdel.obj⏎ --인수를 4개 써보자.
 Volume in drive Chas no label
 Directory of C:\

PCWC   C      PCWC   EXE    PCDEL  C
PCDEL  EXE    TYPEN  EXE
       5 File(s) 1615235 bytes free -- 9개 화일에서 4개가 삭제되었다.
C>
```

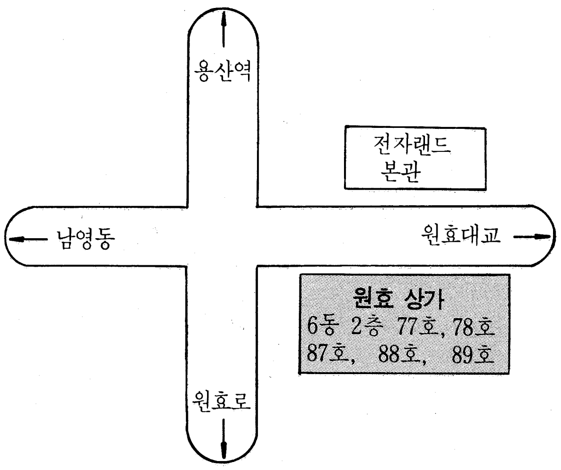
정보사회의 개척자

하나 컴퓨터

이 전 안 내

귀사 및 귀하의 무궁한 발전을
기원합니다.
당사가 본사를 아래와 같이
이전하였습니다.
많은 지도 편달을 바랍니다.

※ 전화 및 FAX, 사서함은 변경없습니다.



대표전화 : 717－4894　　FAX : 715－1546
사 서 함 : 남서울우체국 사서함 47
주　　소 : 서울시 용산구 원효로3가
원효상가 6동 2층 77, 78, 87, 88, 89호

신제품 안내

Top 286 Series

Model : Top 286 A B C D

속　도	12MHz(Zero Wait)
HDD	20MB
FDD	1.2MB
Memory	1MB
I/O	2Serial, 2Parallel 1Joy Port
그래픽	한글, 한자

신제품 개발과 함께
대량생산 체제를 구축하여
수요를 충족시키는 원활한 공급이
가능해졌습니다.
서울지역 및 전국 지역의
대리점 및 딜러를
모집하고 있습니다.

문의 : 본사 대리점 사업부(Tel:717－4894)
대구 총판 (Tel: [053] 425－2318)

비트에 관한 명령과 반복처리 기법

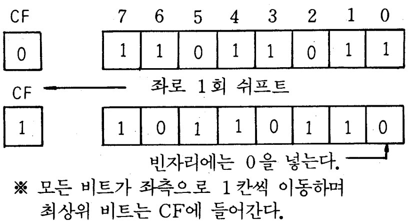
박승제 / 여주컴퓨터학원장

어셈블리는 자료를 바이트단위 혹은 워드단위로 처리하는 것을 기본으로 하고 있지만 임의의 자료에 대해 특정한 비트를 대상으로 조작하는 처리도 가능하다. 비트에 관한 명령은 비트이동 명령과 논리연산 명령으로 나누어지며, 비트이동 명령으로는 쉬프트와 로테이트가 있다.

쉬프트(Shift)는 8비트 혹은 16비트의 자료를 왼쪽 혹은 오른쪽으로 1비트씩 밀치는 것을 말하며, 로테이트(Rotate)는 최상위 비트와 최하위 비트를 연결하여 우측 혹은 좌측으로 회전하는 것을 말한다. 이번 호에서는 쉬프트와 로테이트 등의 비트에 관한 명령과 반복처리를 쉽게 하는 방법에 대해 학습하기로 한다.

쉬프트

쉬프트는 좌로 밀치느냐 우로 밀치느냐에 따라 '좌쉬프트'와 '우쉬프트'로 구별된다. 좌측으로 쉬프트하는 경우에 밀려나온 비트는 CF(캐리 플래그)에 들어가게 되며, 맨 아래 비트는 1로 채워진다. 아래 그림은 8비트의 자료를 좌측으로 1회 쉬프트한 결과를 표시한 것이다.



※ 모든 비트가 좌측으로 1칸씩 이동하며
최상위 비트는 CF에 들어간다.

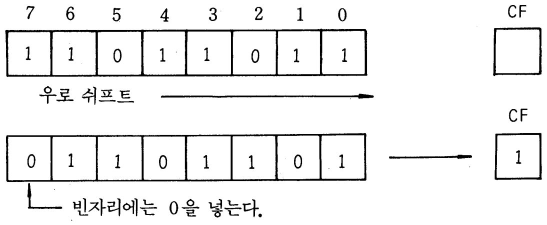
우측으로 쉬프트하는 경우에는 원래의 자료를 부호가 없는 수로 보는 경우와 부호가 붙은 수치로 보는 경우의 2가지로 나누어서 처리를 하게 된다. 부호가 없는 경우에는 좌쉬프트와 같이 밀려나온 비트는 CF에 들어가며, 빈 자리에는 0을 채운다. 부호가 붙은 수치로 보는 경우에는 밀려나온 비트는 CF에 들어가지만 빈 자리에는 최상위 비트(부호 비트 : 1이면 음수, 0이면 양수) 값으로 채우게 된다.

쉬프트는 무조건 0을 채우는 경우를 논리 쉬프트(Logical Shift), 부호 비트로 채우는 경우를 산술 쉬프트(Arithmetic Shift)로 나누어서 부른다(그림 1 참조). 좌측으로 쉬프트하는 경우에는 산술 쉬프트나 논리 쉬프트의 결과가 같다. 또한 자료가 양수일 경우에는 논리 쉬프트나 산술 쉬프트의 결과가 같게 된다.

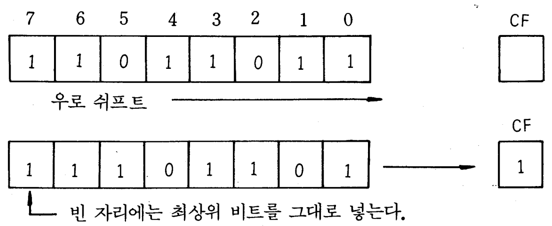
좌로 1회 쉬프트하는 것은 CF를 고려해서 생각하면 자료를 2로 곱한 결과와 같다. 따라서 2회, 3회… 쉬프트할 때마다 4, 8, 16…으로 곱한 결과가 얻어진다. 또한 우로 쉬프트할 경우에는 자료를 2, 4, 8…로 나눈 몫을 얻을 수 있게 된다. 실제의 처리에 있어서 쉬프트는 곱하는 명령(MUL)이나 나누는 명령(DIV)보다 처리 속도가 더 빠르다.

쉬프트에 관한 명령의 형식과 종류는 다음과 같다.

〔그림 1〕 논리쉬프트와 산술쉬프트



(a) 논리 쉬프트



(b) 산술 쉬프트

[형식] 쉬프트 형태 〈레지스터/메모리〉, 〈쉬프트 횟수〉

쉬프트 횟수는 반드시 1이거나 CL 레지스터로 지정해야 한다. 쉬프트형태는 다음과 같다.

SHL : 좌로 쉬프트(논리 쉬프트)
SHR : 우로 쉬프트(논리 쉬프트)
SAL : 좌로 쉬프트(산술 쉬프트)
SAR : 우로 쉬프트(산술 쉬프트)

L : Left
R : Right
SH : 논리 쉬프트
SA : 산술 쉬프트

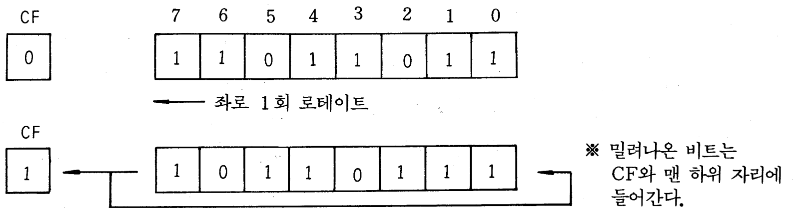
쉬프트의 횟수는 직접 정하거나, 레지스터에 횟수를 기억시켜둔 후 CL레지스터를 이용하기도 한다. 예를 들면 다음과 같이 쓰인다.

SHL AH, 1 : AH의 내용을 좌측으로 1회 쉬프트한다.

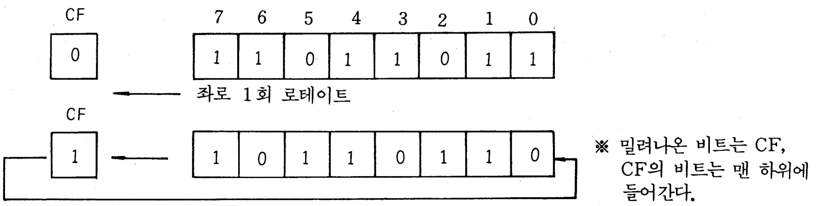
SAR AX, CL : AX의 내용을 CL의 값만큼 우측으로 산술 쉬프트한다.

SHL WORD PTR [BX], 1 : BX가 가리키는 번지의 내용(2바이트)을 1회 좌측으로 쉬프트한다(SAR명령과의 차이점에 유의하여야 한다).

〔그림 2〕 8비트 자료의 로테이트



(a) 캐리를 포함하지 않는 로테이트



(b) 캐리를 포함하는 로테이트

로테이트

로테이트는 상위비트와 하위비트를 연결해서 좌 혹은 우로 회전하는 명령이다. 이 과정에서 밖으로 밀려나온 비트는 빈 자리에 들어가지만 CF에도 들어가게 된다. 또 회전하려는 자료와 캐리플래그(CF)를 연결해서 회전하기도 한다. 그림 2는 8비트의 자료를 좌로 1회 로테이트시킨 과정을 표시한 것이다.

로테이트 명령의 형식은 쉬프트의 경우와 동일하며 OP코드만 다르다. 로테이트의 OP코드는 다음과 같다.

ROL : 좌로 회전
ROR : 우로 회전
RCL : 캐리를 포함하여 좌로 회전
(C : Carry Flag)
RCR : 캐리를 포함하여 우로 회전

쉬프트와 로테이트 명령은 곱셈과 나눗셈을 대신하기도 하고, 자료의 2진수 패턴을 조사하는 데도 쓰인다. 그러면 간단한 예제를 가지고 쉬프트와 로테이트 명령의 쓰임새를 익혀보기로 하자.

[예제 1]
2바이트의 16진수를 입력하여 2진수로 바꾸어서 출력하는 프로그램을 작성하라.

〈처리방법〉

2바이트의 자료는 4문자를 입력받아야 한다. 이들을 2바이트의 수치로 변환한 다음 비트패턴을 조사하게 된다. 비트패턴을 조사하는 방법은 2바이트의 수치를 좌측으로 1회 쉬프트한다. 그러면 밀려나온 비트는 캐리 플래그에 들어갈 것이다. 캐리 플래그가 1이면 화면에 '1'을, 0이면 화면에 '0'을 출력하게 된다. 위의 과정을 16회 반복하면 2바이트의 자료를 2진수 형태로 바꾸어 출력한 결과가 된다.

이때 2바이트를 입력받을 경우 입력하는 방법에 따라 1~4 문자가 되므로 적절히 처리해야 할 것이다. 이를테면 'FF'를 입력했을 경우에는 '00FF'가 입력된 것으로 처리해야 되는 것이다.

위의 처리는 말로는 쉽지만 막상 프로그램을 작성할 경우에는 상당히 번거롭다. 먼저 4문자를 저장할 영역을 메모리상에 설정한 다음 그 부분을 모두 '0'(문자 '0'을 말한다)으로 채운다. 다음에는 입력받은 문자열의 길이를 구한 다음 문자열의 맨 우측부터 차례대로 미리 '0'으로 초기화 시켜놓은 영역의 우측으로 이동시킨다. 이렇게 하면 문자열의 선두에 필요한 만큼의 '0'을 붙일 수 있다. 위의 과정을 도식적으로 표시하면 **그림 3**과 같다.

그림 3과 같이 하면 입력된 문자 앞에 자동적으로 '0'을 부여할 수 있다. 이때 메모리에서 메모리로 직접적인 전송은 불가능하므로 일단은 레지스터를 중계하여야 한다. **리스트** 1의 번지를 간접 지정하는 방식을 잘 살펴보기 바란다. (이 다음 장을 보면 메모리에서 메모리로 직접적인 전송도 가능하므로 자신이 직접 적절히 고쳐보도록 하기 바란다.)

이렇게 입력된 문자열은 좌측부터 2문자씩 끊어 16진수로 변환하게 된다.

변환하는 과정은 지난호에서 다루어 보았으므로 여기서는 설명을 생략한다. 4문자로 2바이트의 16진수를 만들었으면 그 수치를 쉬프트하여 출력한다.

여기서는 매 4비트마다 1칸씩 띄어서 알아보기 쉽게 처리하였다. 리스트의 매행마다 자세히 주석을 붙여두었으므로 별로 어렵지는 않을 것이다.

또 여기서 소문자는 고려하지 않도록 했기 때문에 반드시 '0'~'F'의 대문자를 입력해야 한다. 또한 수치로 볼 수 없는 문자가 입력되었을 경우에는 비프(beep)음을 울리고 다시 입력을 받도록 했다. 비프음은 아스키 코드 7번 문자를 화면에 출력하면 문자가 나타나는 대신 비프음이 발생하게 된다.

논리연산

논리 연산이란 8비트, 혹은 16비트의 자료를 비트단위로 연산하는 것을 말하는데 여기에는 NOT, AND, OR, XOR, NEG, TEST 등이 있다. 이들은 대부분 〈제1 오퍼랜드〉와 〈제2 오퍼랜드〉를 가지고 있다. 〈제1 오퍼랜드〉는 8/16비트의 레지스터이거나 8/16비트의 메모리이며, 〈제2 오퍼랜드〉는 8/16비트의 레지스터이거나 8/16비트의 메모리, 혹은 8/16비트의 수치이다. 그러므로 메모리와 레지스터 혹은 레지스터와 메모리끼리는 연산이 가능하지만 메모리끼리나 수치끼리의 연산은 불가능하다.

(1) NOT

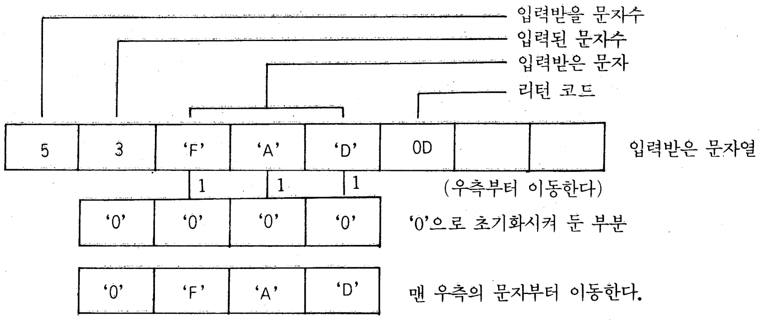
[형식] NOT 〈제1 오퍼랜드〉

[기능] 모든 비트를 반전한다.
(0은 1, 1은 0으로 된다.)

NOT 연산은 1의 보수를 구하는 데 쓰인다. (비트를 반전하는 것을 1의 보수를 구한다고 표기하기도 한다.)

NOT	1	0	1	0	0	1	1	1
	0	1	0	1	1	0	0	0

〔그림 3〕 문자열의 선두를 '0'으로 채우기



〔리스트 1〕 번지의 간접지정

```
CODE    SEGMENT
        ASSUME CS:CODE, DS:CODE
        MOV     AX, CS   ;  data 세그먼트의 설정
        MOV     DS, AX
        JMP     START
MESS    DB      'Input Data : $'
MESS1   DB      'Illegal Data !',0DH,0AH,'$'
WORK1   DB      4 DUP(?)    ;입력한 문자열을 우측으로 맞추어 저장될 영역
WORK2   DW      0           ; 문자열을 16진수로 바꾸어 저장할 영역
DATA    DB      5           ; 입력 받을 문자수(리턴키 포함)
        DB      10 DUP(0)   ; 입력 받을 문자열이 저장되는 영역
 START:
        MOV     DX, OFFSET  MESS  ; 화면에 메세지 출력
        MOV     AH, 9H
        INT     21H
        MOV     DX, OFFSET DATA    ;  문자열 입력
        MOV     AH, 0AH
        INT     21H
        MOV     AL, DATA+1         ; 입력 받은 문자수를 구한다.
        CMP     AL, 0              ; 입력된 문자가 없으면 실행 종료
        JZ      QUIT
        CALL    CLEAR              ; 문자가 저장될 영역을 '0'으로 초기화
        CALL    LOAD               ; 문자열을 꺼내서 우측으로 정열
        CALL    CONVERT            ; 문자열을 16진수로 변환
        JC      AGAIN              ; 변환시 범위 밖의 문자일 경우 처리
        CALL    PRINT              ; 16진수를 2진수로 바꾸어 출력
        JMP     START
 AGAIN: CALL    LINE_FEED
        MOV     AH, 2
        MOV     DL, 7             ; Beep음을 울린다.
        INT     21H
        MOV     DX, OFFSET MESS1 ;  'Illegal Data' 출력
        MOV     AH, 9
        INT     21H
        JMP     START
 QUIT:
        MOV     AH, 4CH           ; 프로그램의 종료
        INT     21H
```

```
CLEAR      PROC                    ; 문자열이 우측을 기준으로 정열될 영역을
        MOV     CL, 4              ; '0'으로 채운다.
        MOV     BX, OFFSET WORK1
        MOV     AL, '0'
   CL1: MOV     [BX], AL
        INC     BX
        DEC     CL
        JNZ     CL1
        RET
CLEAR   ENDP
LOAD       PROC
        MOV     CL, DATA+1          ; 입력된 문자수를 구한다
        MOV     CH, 0
        MOV     BP, OFFSET DATA+1   ; 문자열이 저장된 번지
        MOV     BX, OFFSET WORK1+3  ; 문자열이 새로 저장될 번지
    LP: MOV     SI, CX
        MOV     AL, [BP+SI]         ; 문자를 우측부터 꺼내 이동시키기 위해
        MOV     [BX], AL            ; 서는 번위를 구해야 한다.
        DEC     BX
        DEC     CX
        JNZ     LP
        RET
LOAD    ENDP
CONVERT     PROC
        MOV     BX, OFFSET WORK1        ; 새로 저장된 영역에서 2문자를
        MOV     AH, [BX]                ; 꺼내서 16진수로 바꾼다.
        MOV     AL, [BX+1]
        CALL    CON1
        MOV     BYTE PTR WORK2+1, AH    ; 위에서 바뀐 문자를 상위 바이트로
        MOV     AH, [BX+2]              ; 간주하여 메모리에 격납한다.
        MOV     AL, [BX+3]
        CALL    CON1                    ; 다음의 2문자를 변환 후 하위 바이
        MOV     BYTE PTR WORK2, AH      ; 트로 간주하여 처리
        RET
CONVERT ENDP
CON1        PROC
        CLC                        ; 첫 문자를 16진수 4비트로 바꾼다.
        CMP     AH, '0'            ; 문자가 범위 밖일 경우에 캐리 플래그
        JB      CARRY              ; 를 세트하고 Main으로 돌아가게 된다.
        CMP     AH, '9'
        JA      C1
        SUB     AH, 48
        JMP     C2
    C1: CMP     AH, 'A'
        JB      CARRY
        CMP     AH, 'F'
        JA      CARRY
        SUB     AH, 55
    C2:
        CMP     AL, '0'            ; 둘째 문자를 16진수 4비트로 바꾼다.
        JB      CARRY
        CMP     AL, '9'
        JA      C3
        SUB     AL, 48
        JMP     C4
    C3: CMP     AL, 'A'
        JB      CARRY
        CMP     AL, 'F'
        JA      CARRY
        SUB     AL, 55
    C4: MOV     CL, 4              ; 첫 4비트를 좌로 4비트 비켜서
        SHL     AH, CL             ; 하위 4비트와 더한다.
        ADD     AH, AL
        CLC                        ; CF를 0으로 하여 정상적으로 끝났음을
        RET                        ; 알리고 자신을 호출한 'CONVERT'로 복귀
 CARRY: STC                        ; CF를 1로 하여 스택에서 2바이트를 버리고
        RET -2                     ; 복귀('CONVERT'로 복귀하지 않고 곧바로
CON1    ENDP                       ; Main으로 복귀한다.)
PRINT      PROC
        CALL    LINE_FEED
        MOV     BX, WORK2          ; 위에서 처리한 자료를 BX에 넣는다.
        MOV     CL, 1
   PR1: SHL     BX, 1              ; 좌로 1회 쉬프트
        JC      PR2
        MOV     DL, '0'            ; 캐리가 나타나지 않으면 DL에 '0'을
        JMP     PR3                ; 넣는다.
   PR2: MOV     DL, '1'            ; 캐리가 나타날 때는 '1'을 넣는다.
   PR3: MOV     AH, 2              ; 1문자 화면 출력
        INT     21H
        INC     CL                 ; 쉬프트 회수 증가
        CMP     CL, 5              ; 매 4비트를 처리한 후에 1칸을 띄운다.
        JE      PR4
        CMP     CL, 9
        JE      PR4
        CMP     CL, 13
        JE      PR4
        CMP     CL, 16             ; 16회를 처리하지 않았으면 위의 과정을 반복
        JBE     PR1
        CALL    LINE_FEED
        JMP     PR5
   PR4: MOV     DL, ' '
        INT     21H
        JMP     PR1
   PR5: RET
PRINT   ENDP                       ; 출력 후에 줄을 바꾸는 처리 루틴
LINE_FEED  PROC
        MOV     AH, 2
        MOV     DL, 0DH
        INT     21H
        MOV     DL, 0AH
        INT     21H
        RET
LINE_FEED  ENDP
CODE       ENDS
           END
```

(2) AND

[형식] AND 〈제1 오퍼랜드〉, 〈제2 오퍼랜드〉

[기능] 비트의 논리곱을 취한다.

AND의 연산은 2개의 비트가 1일 경우에만 결과가 1이 되고 그 외에는 0이 된다.

	1	0	1	0	1	1	0	1
AND	0	0	0	0	1	1	1	1
	0	0	0	0	1	1	0	1

AND연산은 특정 비트를 0으로 하는 경우에 쓰인다. 위의 연산은 상위 4비트의 값을 모두 0으로 하는 보기이다. 이때 0으로 하고 싶은 비트의 값은 0, 그 외에는 1의 값을 가진 자료와 연산을 하면 된다.

(3) OR

[형식] OR 〈제1 오퍼랜드〉,
〈제2 오퍼랜드 〉

[기능] 비트의 논리합을 취한다.

OR의 연산은 2개의 비트가 0일 경우에만 결과가 0이 되고 그 외에는 1이 된다. OR의 연산은 특정한 비트를 1로 세우는데 이용된다. 이 경우에는 1로 하고 싶은 부분의 비트1, 그외는 0의 값을 갖는 자료와 OR연산을 시키면 된다.

	1	0	0	1	1	0	1	0
OR	1	1	1	1	0	0	0	0
	1	1	1	1	1	0	1	0

위의 연산의 결과 상위 4비트가 모두 1이 된다.

(4) XOR

[형식] XOR 〈제1 오퍼랜드〉,
〈제2 오퍼랜드〉

[기능] 비트의 배타적 논리합을 취한다.

배타적 논리합(Exclusive OR)은 2개의 비트가 모두 같을 경우에는 0, 그 외에는 1의 값을 갖는다. 다시 말하면 어느 한쪽만 1일 때 1의 값을 갖고 그 외에는 모두 0이 된다. XOR은 특정 비트를 반전하거나 레지스터의 내용을 0으로 클리어시키는데 쓰인다. 이때 반전하려는 부분은 1, 그외에는 0의 값을 가진 자료와 XOR 연산을 하면 된다.

	1	1	0	1	1	0	1	0
XOR	1	1	1	1	0	0	0	0
	0	0	1	0	1	0	1	0

위의 연산을 한 결과 상위 4비트의 내용이 반전되었다. 또, 값이 같은 자료끼리 XOR연산을 하면 그 결과는 항상 0이 된다.

```
MOV AX, 0
XOR AX, AX
```

위의 명령은 서로 결과가 같다. (AX가 0이 된다) 그러나 수행속도는 후자의 경우가 더 빠르다. 그러므로 XOR 명령은 레지스터의 내용을 0으로 클리어 시키는데 자주 이용된다.

(5) NEG

[형식] NEG 〈제1 오퍼랜드〉

[기능] 2의 보수를 취한다.

NEG연산은 부호를 바꾸는 구실을 한다. 즉, 양수는 음수로, 음수는 양수로 바꾼다. 컴퓨터에서 부호를 바꾸는 것을 2의 보수를 취한다고 한다.

정수로 음수를 표현하는 방법에 있어 '부호와 그 절대값에 의한 방법', '1의 보수에 의한 방법', '2의 보수에 의한 방법' 등이 있는데 요즈음 거의 모든 컴퓨터에서는 2의 보수에 의한 방법으로 음수를 표현하고 있다. 그러므로 2의 보수를 취하는 것은 먼저 1의 보수를 구한 다음 그 값에 1을 더하면 된다.

1의 보수를 구한다는 의미는 비트를 반전하는 것과 같다.

NEG	0	1	0	1	1	0	1	0
	1	0	1	0	0	1	1	0

위의 연산은 5AH의 부호를 바꾸어 −5AH로 만든다.

(6) TEST

[형식] TEST 〈제1 오퍼랜드〉,
〈제2 오퍼랜드〉

[기능] 〈제1 오퍼랜드〉의 특정 비트 패턴을 조사한다.

TEST 명령은 〈제1 오퍼랜드〉와 〈제2 오퍼랜드〉사이에 AND 연산을 수행한 후 그 비트가 1일 경우에 CF를 1로 세트한다. 이때 〈제1 오퍼랜드〉의 값은 변하지 않는다. 이때 〈제2 오퍼랜드〉는 단 1개의 비트만 1로 되어 있어야 한다. 다음의 TEST 명령은 최상위 비트의 비트 패턴을 조사하는 보기이다.

	1	1	0	1	1	0	1	0
TEST	1	0	0	0	0	0	0	0

조사하려는 부분만 1로 한 값
원래의 데이터

위의 경우 최상위 비트끼리 AND 연산을 하면 그 비트가 1이 되므로 CF의 값이 1이 된다.

반복 처리 명령

8088에는 비교와 분기를 합쳐서 하나로 하는 명령이 있다.

LOOP 명령과 스트링(string) 명령이 바로 그것이다.

루프 명령과 스트링 명령을 이용하면 반복 처리를 쉽게 할 수 있다.

(1) LOOP 명령

[형식] LOOP 〈레이블〉
[기능] CX의 값을 1 감소하고 그 값이 0이 아니면 〈레이블〉로 분기한다.

LOOP 명령은 CX 레지스터를 뺄셈식 카운터로 사용하며 CX가 0이 될 때까지 〈레이블〉이 붙은 곳을 반복 처리한다. 이런 절차를 우리가 알고 있는 명령으로 처리하면 다음과 같다.

```
레이블:
        ......
        DEC  CX
        CMP  CX, 0
        JNE  레이블
```

위의 4행을 좀 더 줄이면 다음과 같다.

```
레이블:
        ......
        DEC  CX
        JNZ  레이블
```

LOOP명령을 사용하면 다음과 같다.

```
레이블:
        ......
        LOOP  레이블
```

LOOP 명령에서 사용하는 카운터는 CX 레지스터로 정해져 있으며, LOOP 명령을 사용하기 전에 CX에 반복횟수를 지정해야 하는 것을 명심해야 한다.

또한 LOOP명령은 CX의 값이 0이 되기까지 무조건 반복하는 것외에 어떤 조건을 주어 반복 여부를 결정하는 방법도 가지고 있다. 여기서의 조건이란 연산결과 후 제로플래그(Zero Flag)가 나타났는지의 여부로 결정한다. 그러므로 연산 결과가 0이냐 0이 아니냐를 나타내는 'Z'와 'NZ'를 'LOOP'에 붙여서 결정한다.

'LOOPZ' 명령은 연산결과가 0이고 CX의 값이 0이 아니라면 반복처리를 한다. 또, 'LOOPNZ' 명령은 연산 결과가 0이 아니고 CX의 값도 0이 아니면 반복 처리를 하고 그 외에는 루프를 벗어나게 된다. 이와 같은 조건부 LOOP문은 반복 상한선을 CX로 지정해두고 반복하는 동안 어떤 조건이 만족하면 반복을 중단하는 경우에 흔히 이용된다.

우리가 어떤 자료 A와 B의 크기를 비교하는 경우 그 결과는 다음의 3가지 중의 하나가 된다.

① A=B
② A〈B
③ A〉B

컴퓨터에서는 A와 B의 크기를 비교하기 위해서는 A에서 B를 뺀 결과를 가지고 크기를 판단하게 된다. A−B의 결과가 양수이면 A〉B이며, 음수이면 A〈B이다. 또 결과가 0이면 A=B이다.

어셈블리에서는 연산의 결과가 0이면 제로 플래그가 나타난다는 것은 이제 독자들은 익히 알고 있는 사실일 것이다. 그러므로 연산 그 결과가 0이 되었다는 것은 A와 B가 같다(Equal)는 의미이며, 또 달리 표현하면 제로(Zero) 플래그가 나타났다고도 표현할 수 있다. 그러므로 어셈블리에서는 '같다'는 의미의 'E'와 제로 플래그가 나타났다는 'Z'를 같은 것으로 취급한다. 따라서 위의 조건부 LOOP명령은 LOOPZ, LOOPNZ 대신에 LOOPE, LOOPNE라는 표현도 가능하게 되며 실제로 8088 어셈블러는 같은 것으로 취급한다.

이때 어떤 표기법이 좋으냐 하는 것은 프로그래머의 취향에 따라 결정되게 된다. 그러면 예제를 가지고 LOOP 명령을 실습해보기로 하자.

> [예제 2]
> 임의의 문자열을 입력받아 선두의 공백을 제거하여 출력하되, 소문자는 모두 대문자로 바꾸어서 출력하는 프로그램을 작성하라.

〈처리 방법〉

문자열의 입력은 DOS 기능호출의 0AH번을 이용하기로 하자. 문자열의 선두부터 한 문자씩 꺼내 그 문자가 공백이 아니라면 그 이후의 문자열을 화면에 출력하기로 하고, 공백이라면 다음 문자를 판단하는 과정을 반복하게 된다(리스트 2 참조). 소문자를 대문자로 바꾸는 방법은 다음과 같다.

① 1문자를 추출한다. 이 문자가 'a'~'z' 사이의 문자가 아닐 경우에는 변환을 하지 않는다.
② 위의 문자에 'a'의 아스키 코드에서 'A'의 아스키 코드의 차이를 뺀다. 'a'에서 'A'를 뺀 값은 아스키 코드를 보고 구할 수도 있지만 대부분의 어셈블러에서는 수식을 지정하면 자동으로 계산하므로 다음과 같이 기술하면 된다.

```
SUB AL, 'a'-'A'
```

위의 명령은 'a'에서 'A'를 뺀 값을 AL에서 뺄셈하게 된다.

③ 입력된 문자열의 끝에는 0DH가 들어가므로 이 문자까지 출력했을 경우에는 라인피드를 시키고 출력을 마친다.

(2) 스트링 명령

스트링 명령은 메모리에 연속적으로 저장된 여러 바이트의 자료들을 전송 및 비교하는 명령을 통틀어서 부르는 것으로, 리피트 프리픽스(Repeat Prefix) 라는 명령과 조합해서 대량의 데이터를 간단하게 처리해주는 명령이다. 또한 이들의 명령은 메모리에서 메모리로 직접 데이터를 전송해주는 처리체계도 가지고 있으므로 대량의 데이터를 이동하는 경우에 아주 유용하게 쓰인다. 스트링 명령은 다음과 같은 것들이 있으며 이들은 오퍼랜드를 가지지 않는다.

① LODSB : 메모리에서 AL로 1바이트 전송
② LODSW : 메모리에서 AX로 1워드 전송

③ MOVSB : AL에서 메모리로 1바이트 전송
④ MOVSW : AX에서 메모리로 1워드 전송
⑤ SCASB : 메모리와 AL의 비교
⑥ SCASW : 메모리와 AX의 비교
⑦ CMPSB : 메모리와 메모리끼리의 비교(바이트 단위)
⑧ CMPSW : 메모리와 메모리끼리의 비교(워드 단위)

스트링 명령에는 메모리의 번지를 지정하기 위해서 특별한 레지스터를 이용하며 데이터를 읽어낼 때와 쓸 때 주소를 지정하는 레지스터가 각각 다르다.

① 데이터를 읽어낼 때

DS 세그먼트의 SI 레지스터가 가리키는 번지를 대상으로 한다. 즉, DS:[SI]의 내용이 된다.

② 데이터를 쓸 때

ES 세그먼트의 DI 레지스터가 가리키는 번지를 대상으로 한다. 즉, ES:[DI]에 쓰게 된다.

그러므로 스트링 명령을 사용하기 위해서는 반드시 프로그램의 첫머리나 필요한 부분에서 DS와 ES의 설정을 하여야 한다. 이것은 동일 세그먼트내에서 처리를 하는 경우에도 꼭 필요한 부분이다. 보통은 다음과 같이 한다.

```
MOV  AX, CS
MOV  DS, AX
MOV  ES, AX
```

또한 읽어들이거나 써넣기 위해서는 반드시 AL레지스터나 AX레지스터를 이용해야 한다. 또한 스트링 명령은 수행 후에 자동적으로 SI나 DI레지스터의 값을 1 혹은 2씩 증가시키게 된다. 처리하는 자료가 바이트 단위이면 1, 워드단위이면 2씩 증가하게 된다.

이때 디렉션 플래그의 값이 1이면 SI, DI의 값은 1 혹은 2씩 감소하게 된다. 그러므로 메모리의 하위번지부터 상위번지로 거슬러 올라가며 처리를 해야 하는 경우에는 디렉션 플래그의 값을 0으로 하고, 반대로 상위 번지에서 하위 번지로 내려가며 처리를 해야하는 경우에는 디렉션 플래그의 값을 1로 해두면 된다.

디렉션 플래그의 값을 1로 하느냐 0으로 하느냐 하는 것은 프로그래머가 지정하는 것으로 캐리 플래그를 0 혹은 1로 설정하는 방법과 유사하며 다음과 같은 명령이 있다.

CLD : 디렉션 플래그를 0으로 클리어(Clear) 한다.
STD : 디렉션 플래그를 1로 세트(Set).

〔리스트 2〕 소문자를 대문자로 치환

```
CODE      SEGMENT
          ASSUME  CS:CODE, DS:CODE, ES:CODE
          MOV   AX, CS
          MOV   DS, AX
          MOV   ES, AX
          JMP   MAIN
MESS      DB    'Input String : $'
DATA      DB    255
          DB    258 DUP (' ')
MAIN:     MOV   AH, 9
          MOV   DX, OFFSET MESS
          INT   21H
          MOV   AH, 0AH
          MOV   DX, OFFSET  DATA
          INT   21H
          CALL  LINE_FEED
          MOV   BX, OFFSET DATA
          MOV   AL, [BX+1]
          CMP   AL, 0
          JZ    QUIT
          CALL  PRINT_STR
          JMP   MAIN
  QUIT:   MOV   AH, 4CH
          INT   21H
PRINT_STR     PROC
          MOV   AH, 2
          MOV   CX, 255
PRT1:     INC   BX
          MOV   DL, [BX+1]
          CMP   DL, ' '
          LOOPZ PRT1
PRT2:     MOV   DL, [BX+1]
          CMP   DL, 'a'
          JB    PRT3
          CMP   DL, 'z'
          JA    PRT3
          SUB   DL, 'a'-'A'
 PRT3:    INT   21H
          INC   BX
          CMP   DL, 0DH
          LOOPNZ PRT2
          CALL  LINE_FEED
          RET
PRINT_STR     ENDP
LINE_FEED     PROC
          MOV   AH, 2
          MOV   DL, 0DH
          INT   21H
          MOV   DL, 0AH
          INT   21H
          RET
LINE_FEED     ENDP
CODE      ENDS
          END
```

REP 명령

REP는 리피트 프리픽스라고 부르며 스트링 명령의 앞에 기술하여 스트링 명령의 반복을 지시하는 명령이다. 이때 반복횟수는 미리 CX 레지스터에 넣어두어야 한다. REP명령은 LOOP 명령과 마찬가지로 CX 레지스터를 뺄셈식 카운터로 이용하며 CX의 값이 0이 될 때까지 스트링명령을 반복시킨

```
REP   MOVSB
```

위의 명령은 DS:[SI]번지의 내용을 ES:[DI]번지로 CX의 값만큼 블록 전송하게 된다. 위의 명령을 LOOP 명령으로 바꾸면 다음과 같다.

```
NEXT:
    MOVSB
    LOOP  NEXT
```

위의 명령을 스트링 명령을 이용하지 않고 기술하면 다음과 같다.

```
NEXT:
    MOV   AL, DS:[SI]
    MOV   ES:[DI], AL
    INC   SI
    INC   DI
    DEC   CX
    JNZ   NEXT
```

리피트 프리픽스 명령은 LOOP 명령과 마찬가지로 조건에 의한 반복처리도 가능하다. 여기서 조건이라는 것은 연산결과 후 0이 되었느냐 (제로 플래그가 나타났느냐)는 것을 말한다.

그러므로 'REPZ' 혹은 'REPNZ'라는 명령도 있으며, 'Z' 대신에 'E'를 쓸 수도 있다.

이와 같은 명령은 메모리의 내용을 검색한다든지, 서로 비교하는 경우에 비교할 상한선을 CX에 지정해 두고 같은 문자를 찾는다든지, 혹은 2개의 문자열에서 일치하지 않는 문자를 발견하려는 경우에 많이 쓰인다. 그러면 예제를 가지고 스트링 명령의 쓰임을 익혀보기로 하자.

〔예제 3〕
16진수 주소를 입력받아 그 번지부터 320바이트 분을 화면에 덤프(Dump)하는 프로그램을 작성하여라.

〈처리 방법〉

디버거 등에서 처리하는 메모리 덤프 프로그램을 작성하는 보기이다. 이 정도의 프로그램을 스스로 작성할 수 있다면 어셈블리 실력이 중급은 되는 셈이다.

이 처리는 먼저 16진수 4자리(1~4문자)를 입력받아 2바이트의 16진수를 바꾸어 번지를 계산하는 부분과, 그 부분부터 320바이트를 화면에 덤프하는 부분으로 나누어 생각할 수 있다.

번지로 변환하는 부분은 리스트 1에서 사용한 방법을 그대로 이용하면 되겠고, 메모리의 내용을 1바이트 꺼내와서 그것을 상위 4비트와 하위 4비트로 나누어 16진수 문자로 바꾸어 출력하는 것도 지난호의 강좌에서 다루어 보았으므로 별도 어려울 것은 없을 것이다.

〔그림 4〕 AL레지스터의 내용을 2문자로 바꾸는 과정

① AL의 자료를 AH로 옮긴다.(MOV AH, AL)

	0	1	1	0	1	1	0		AL	(6CH)
	0	1	1	0	1	1	0	0	AH	(6CH)

② AH의 값을 우측으로 4회 쉬프트한다.(MOV CL, 4 : SHR AL, CL)

	0	0	0	0	0	1	1	0	AH	(06H)

상위 4비트가 하위 4비트가 된다.

③ 이 값이 0~9 사이의 값이면 48을, 9 이상의 값이면 55를 더한다.
이 수치가 상위 4비트를 1문자로 바꾼 값이다.

	0	0	0	0	0	1	1	0	AH	(06H)
+	0	0	1	1	0	0	0	0	48	(30H)
	0	0	1	1	0	1	1	0		

④ AL의 값을 0FH와 AND연산을 하여 상위 4비트를 0으로 클리어 한다.

	0	1	1	0	1	1	0	0	AH	(6CH)
AND	0	0	0	0	1	1	0	0	0FH	
	0	0	0	0	1	1	0	0	AL	(0CH)

⑤ 이 수치에 48, 혹은 55를 더한다.(③의 과정으로 처리)

	0	0	0	0	1	1	0	0	AL	(0CH)
+	0	0	1	1	0	1	1	1	55	(37H)
	0	1	0	0	0	0	1	1	AL(67 : C의 아스키코드)	

메모리의 내용은 화면에 16바이트 단위로 출력을 하도록 하고 그 끝에 체크 섬(Checksum)을 붙이도록 한다. 1바이트의 수치를 2개의 문자로 바꾸는 것은 지난 호에도 다뤄본 적이 있지만 이번 호에서는 다른 방법을 이용해 보기로 한다.

AL레지스터에 6CH가 저장되어 있다고 가정을 하고 이 수치를 '6'과 'C'의 2문자로 바꾸는 과정을 그림 4에 나타내었다.

그림 4의 과정을 거치면 AL의 자료가 AH, AL의 2개의 문자로 바뀌게 되므로 DOS 기능호출 1문자 출력루틴을 이용해서 출력하면 된다. 위의 처리를 리스트 3에 실었다. 필요한 부분은 소스상에 주석을 달아두었으므로 프로그램의 설명은 생략한다.

끝으로

독자들의 성원 속에 시작한 강좌가 벌써 6회가 되었다. 이제는 특수한 명령을 제외하고는 거의 배운 셈이 된다. 어셈블리는 명령어만 알아가지고는 프로그램을 작성하기 어렵다. 명령어도 물론 중요하지만 그보다는 하드웨어에 대한 지식이 있어야 한다. 다음 호에서는 DOS 인터럽트와 BIOS 인터럽트를 총괄해서 인터럽트 리스트를 소개할 예정이다. PC

〔리스트 3〕 1 바이트의 수치를 2 개의 문자로 변경

```
CODE      SEGMENT
          ASSUME CS:CODE, DS:CODE, ES:CODE
          MOV    AX, CS   ;  data 세그먼트의 설정
          MOV    DS, AX
          MOV    ES, AX   ; 엑스트라 세그먼트의 설정
          JMP    START
MESS      DB     'Input Address : $'
MESS1     DB     'Illegal Address !',0DH,0AH,'$'
WORK1     DB     4 DUP(?)   ;입력한 번지를 우측으로 맞추어 저장될 영역
ADDR      DW     0          ;번지를 16진수로 바꾸어 저장할 영역
ADDR_STR  DB     '0000 : $' ;번지를 문자열로 바꾸어 저장할 영역
C_SUM     DB     ': 0000',0DH,0AH,'$' ; 체크섬을 문자열로 바꿔 저장할 영역
DATA      DB     5          ; 입력 받을 문자수(리턴키 포함)
          DB     10 DUP(0)  ; 입력 받을 문자열이 저장되는 영역
START:
          MOV    DX, OFFSET  MESS  ; 화면에 메세지 출력
          MOV    AH, 9H
          INT    21H
          MOV    DX, OFFSET DATA   ;  문자열 입력
          MOV    AH, 0AH
          INT    21H
          MOV    AL, DATA+1        ; 입력 받은 문자수를 꺼낸다.
          CMP    AL, 0             ; 입력된 문자가 없으면 실행 종료
          JZ     QUIT
          CALL   CLEAR
          CALL   LOAD
          CALL   CONVERT           ; 문자열을 16진수로 변환
          JC     AGAIN             ; 변환시 범위 밖의 문자일 경우 처리
          CALL   DUMP              ; 메모리의 내용을 덤프
          JMP    START
  AGAIN:  CALL   LINE_FEED
          MOV    AH, 2
          MOV    DL, 7             ; Beep음을 울린다.
          INT    21H
          MOV    DX, OFFSET MESS1 ;  'Illegal Data' 출력
          MOV    AH, 9
          INT    21H
          JMP    START
 QUIT:
          MOV    AH, 4CH           ; 프로그램의 종료
          INT    21H
          ;
CLEAR        PROC                  ; 문자열이 우측을 기준으로 정열될 영역을
          MOV    CL, 4             ; '0'으로 채운다.
          MOV    BX, OFFSET WORK1
          MOV    AL, '0'
   CL1:   MOV    [BX], AL
          INC    BX
          DEC    CL
          JNZ    CL1
          CALL   LINE_FEED
          RET
CLEAR     ENDP
          ;
LOAD         PROC
          MOV    CL, DATA+1          ; 입력된 문자수를 얻는다
          MOV    CH, 0
          MOV    BP, OFFSET DATA+1   ; 문자열이 저장된 번지
          MOV    BX, OFFSET WORK1+3  ; 문자열이 새로 저장될 번지
   LP:    MOV    SI, CX
          MOV    AL, [BP+SI]         ; 문자를 우측부터 꺼내 이동시키기 위해
          MOV    [BX], AL            ; 서는 번위를 설정한다.
          DEC    BX
          DEC    CX
          JNZ    LP
          RET
LOAD   ENDP
          ;
CONVERT       PROC
          MOV    BX, OFFSET WORK1        ; 새로 저장된 영역에서 2문자를
          MOV    AH, [BX]                ; 꺼내서 16진수로 바꾼다.
          MOV    AL, [BX+1]
          CALL   CON1
          MOV    BYTE PTR ADDR+1, AH     ; 위에서 바뀐 문자를 상위 바이트로
```

```
        ADD     DL, AL              ; 체크섬을 계산
        ADC     DH, 0               ; 자리올리기가 있는가?
        CALL    PRINT_HEX           ; 1바이트를 2문자로 바꿔서 출력
        LOOP    DU2
        MOV     AX, DX              ; 체크섬을 AX로 옮겨서 출력할 루틴을
        CALL    PRINT_SUM           ; 호출
        POP     CX
        LOOP    DU1                 ; 20줄을 출력하지 않았으면 반복처리
        RET
DUMP        ENDP
        ;
PRINT_ADDR  PROC
        MOV     AX, SI              ; 번지를 AX로 옮긴다
        PUSH    AX
        MOV     BP, OFFSET ADDR_STR ; 문자열로 바꾸어 저장할 영역 지정
        MOV     AL, AH              ; 상위 바이트를 2문자로 치환
        CALL    MAKE_CHR
        MOV     [BP], AH
        MOV     [BP+1], AL
        POP     AX                  ; 하위 바이트를 2문자로 치환
        CALL    MAKE_CHR
        MOV     [BP+2], AH
        MOV     [BP+3], AL
        MOV     DX, OFFSET ADDR_STR ; 치환된 문자열의 출력
        MOV     AH, 9
        INT     21H
        RET
PRINT_ADDR  ENDP
        ;
PRINT_HEX   PROC
        PUSH    DX
        CALL    MAKE_CHR    ; AL의 값을 2문자로 바꿀 루틴 출력
        MOV     DH, AL      ; 하위 바이트의 보존
        MOV     DL, AH      ; 상위 바이트 출력
        MOV     AH, 2
        INT     21H
        MOV     DL, DH      ; 하위 바이트 출력
        INT     21H
        MOV     DL, ' '     ; 공백 1문자 출력
        INT     21H
        POP     DX
        RET
PRINT_HEX   ENDP
        ;
PRINT_SUM   PROC
        MOV     BP, OFFSET C_SUM    ; 체크섬을 문자열로 바꾸어 저장될 영역
        MOV     AL, DH              ; 상위 바이트를 2문자로 치환
        CALL    MAKE_CHR
        MOV     [BP+2], AH          ; 문자열 영역에 저장
        MOV     [BP+3], AL
        MOV     AH, [BX+2]              ; 간주하여 메모리에 격납한다.
        MOV     AL, [BX+3]
        CALL    CON1                    ; 다음의 2문자를 변환 후 하위 바이
        MOV     BYTE PTR ADDR, AH       ; 트로 간주하여 처리
        RET
CONVERT ENDP
        ;
CON1        PROC
        CLC                         ; 첫 문자를 16진수 4비트로 바꾼다.
        CMP     AH, '0'             ; 문자가 범위 밖일 경우에 캐리 플래그
        JB      CARRY               ; 를 세트하고 Main으로 돌아가게 된다.
        CMP     AH, '9'
        JA      C1
        SUB     AH, 48
        JMP     C2
    C1: CMP     AH, 'A'
        JB      CARRY
        CMP     AH, 'F'
        JA      CARRY
        SUB     AH, 55
    C2:
        CMP     AL, '0'             ; 둘째 문자를 16진수 4비트로 바꾼다.
        JB      CARRY
        CMP     AL, '9'
        JA      C3
        SUB     AL, 48
        JMP     C4
    C3: CMP     AL, 'A'
        JB      CARRY
        CMP     AL, 'F'
        JA      CARRY
        SUB     AL, 55
    C4: MOV     CL, 4               ; 첫 4비트를 좌로 4비트 비켜서
        SHL     AH, CL              ; 하위 4비트와 더한다.
        ADD     AH, AL
        CLC                         ; CF를 0으로 하여 정상적으로 끝났음을
        RET
 CARRY: STC
        RET -2                      ; 스택에서 2바이트를 버리고 복귀
CON1    ENDP

DUMP        PROC
        MOV     BX, OFFSET ADDR
        MOV     CX, 20              ; 출력할 줄 수 설정
        MOV     SI, [BX]            ; 덤프할 번지 설정
   DU1: PUSH    CX
        CALL    PRINT_ADDR          ; 번지를 출력
        MOV     CX, 16              ; 1 줄당 출력할 바이트 수 설정
        MOV     DX, 0               ; 체크섬을 구할 레지스터를 초기화
   DU2: LODSB                       ; [SI] 의 내용을 1바이트 AL로 로드
        MOV     AL, DL              ; 하위 바이트를 2문자로 치환
        CALL    MAKE_CHR
        MOV     [BP+4], AH
        MOV     [BP+5], AL
```

```
        MOV     DX, OFFSET C_SUM   ; 치환된 체크섬을 출력
        MOV     AH, 9
        INT     21H
        RET
PRINT_SUM    ENDP
        ;
MAKE_CHR     PROC
        PUSH    CX
        MOV     AH, AL          ; AL의 내용을 AH에 복사
        MOV     CL, 4           ; (상위 4비트 처리)
        SHR     AH, CL          ; AH를 우측으로 4비트 쉬프트
        CMP     AH, 9           ; 이 값이 9보다 크면 55를 더하고
        JA      MA1             ; 9이하이면 48을 더한다.
        ADD     AH, 48          ;
        JMP     MA2
   MA1: ADD     AH, 55
   MA2: AND     AL, 0FH         ; AL의 상위 4비트를 0으로 지운다.
        CMP     AL, 9
        JA      MA3
        ADD     AL, 48
        JMP     MA4
   MA3: ADD     AL, 55
   MA4: POP     CX
        RET
MAKE_CHR     ENDP
        ;
LINE_FEED  PROC
        MOV     AH, 2
        MOV     DL, 0DH
        INT     21H
        MOV     DL, 0AH
        INT     21H
        RET
LINE_FEED  ENDP
CODE       ENDS
           END
```

마이크로프로세서 마이크로컴퓨터 (하드웨어 과정) 講義案內 개강 매월 5일 (2개월 완성)

본원에서는 금번 2개월 과정으로 digital circuit 설계로 부터 CPU 및 memory, 주변장치 등의 구조 및 동작원리를 습득하여 computer system 및 주변기기의 Interface를 설계하는 능력을 함양하고자 합니다. 또 이 과정에서는 Z-80계 computer 및 APPLE, IBM PC 등의 system을 대상으로 구성, 동작원리, 프로그래밍 실습을 하여 궁극적으로 MICOM 및 digital system의 설계 및 수리능력을 배양하고자 하오니 많은 참여 바랍니다.

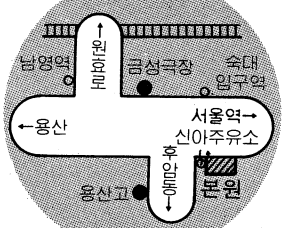
■ 강습일정

주별	일차	강습 내용 (이론 및 실기)	주별	일차	강습 내용 (이론 및 실기)
1주	1일차	computer의 기본 구조와 8bit CPU의 종류 및 내부 구성	5주	1일차	Z80 CPU의 구조 및 Timing 특성
	2일차	논리회로 설계원리 및 그 응용		2일차	Z80 CPU의 Interrupt (IM0, IM1, IM2)
	3일차	마이크로 컴퓨터 시스템의 구성		3일차	μ-Computer의 회로설계 및 실습
2주	1일차	Karnaugh Map을 이용한 디지탈회로 설계	6주	1일차	Assembly Language를 이용한 프로그램 실습
	2일차	Flip Flop 및 Counter 등을 이용한 응용회로 설계		2일차	8255 및 Z80 P10의 특성 및 응용
	3일차	CPU의 주변회로 및 Address Map/Decoding		3일차	16bit MPU(MC68000)의 구조 및 원리
3주	1일차	TTL 논리회로 설계 실습	7주	1일차	Z80 S10의 특성 및 응용
	2일차	A/D 및 D/A Converter 및 ALU		2일차	EPROM Writer 및 Floppy Disk Controller
	3일차	Memory(ROM/RAM/DRAM)의 구성		3일차	MC68000의 Signal 및 Bus Operation
4주	1일차	8080 및 8085 CPU의 구조 및 Timing 특성	8주	1일차	음성합성 장치의 설계
	2일차	8085 Assembly언어 프로그래밍		2일차	Bus Standard, Modem, Data통신 소개
	3일차	I/O(Port, Keyboard etc)의 원리 및 Video Generator		3일차	16bit MPU(MC68000)의 회로 설계

Textbook

1. TTL응용실무
2. Z80 마이크로컴퓨터
3. Microprocessor Interfacing Technique
4. Microprocessor and Digital Systems

Hardware + Software = ?



전자산업 4반세기를 지켜온 기술인의 전당

관인 **한국전자통신학원** 컴퓨터 기술부

(구 한국전파학원)

140 서울시 용산구 갈월동 17번지 ☎ 754-1751/2902/1044

기계어 코드를 간단하게 베이직 데이터로

김대순/아주대 A-MiCOM

필자가 소개하고자 하는 프로그램은 기계어 코드를 베이직 데이터로 변환해주는 기능을 가진 유틸리티이다.

일반적으로 유저가 처음 컴퓨터를 접할 때 익히는 언어는 베이직일 것이다. 필자의 경우도 예외는 아니며, 처음 컴퓨터를 배우고 공부할 때에는 이러한 베이직을 가장 먼저 배웠다. 그런데 베이직 프로그램에서 데이터를 입력할 때는 프로그램의 길이가 길거나 하여 데이터가 많을 땐 입력시 필자로 하여금 얼굴을 찡그리게 해주었다.이런 데이터가 많은 리스트를 잡지에 게재하는 분들은 어떻게 그 많은 데이터를 입력할까? 그 많은 걸 다 입력하려면 유능한 키펀처(?)가 아닌 이상 짜증을 느꼈을텐데… 하는 생각을 하게 되었다. 그러던중 마침 외지에서 필자의 생각을 비웃기라도 하듯이 이에 관련된 기사를 접하게 되어, 필자의 생각이 모자르다는 것을 느끼게 해주었다. 아무리 작은 프로그램이라 하더라도 그 활용에 따라, 마치 명기수를 기다리는 명마가 되기도 하고, 아무 쓸모도 없는 것이 되기도 한다는 것을 느끼게 해준 계기가 되었다.

다음에 소개하고자 하는 프로그램은 그러한 불편을 해소시켜주는 프로그램이다.

먼저 프로그램의 사용방법과 프로그램의 알고리즘에 대해서 언급해보자.

프로그램의 입력과 실행

먼저 리스트 1의 프로그램을 입력하려면 인터프리터 또는 컴파일러(GW-BASIC 또는 QUICK BASIC 등) 방식의 베이직을 기동시켜 리스트 1 BIN2DAT.BAS를 한 라인씩 조심스럽게 입력한다. 다 입력한 후에는 틀리게 입력한 부분이 없는지를 검사하여 프로그램의 오타가 없는 경우에는 이를 안전하게 BIN2DAT.BAS란 화일명으로 세이브시켜 놓는다.

이 프로그램을 실행시키면 먼저 사용자가 변환을 원하는 프로그램의 화일명을 물어본다. 이때 화일명을 입력하고, 리턴키(↵)를 누른다. 그러면 BIN2DAT.BAS라는 특별한 안전장치(?)로서, 입력받은 화일명의 확장명이 COM인가 확인하게 된다. 데이터 변환시에는 반드시 COM이란 확장자를 가져야 하는데, 그 이유는 EXE형 화일에는 디스크상에 프로그램 실행과 직접 연관이 없는 DOS를 위한 헤더라는

〔리스트 1〕 BIN2DAT.BAS 프로그램

```
10 DEF SEG
20 KEY OFF
30 ON ERROR GOTO 570
40 :
50 REM print title and get infor
60 :
70 PRINT "Binary to data statement converter"
80 PRINT "Copyright 1986, Compute! Publications"
90 PRINT
100 INPUT "File to convert: ", FSOURCE$
110 IF INSTR(FSOURCE$,".COM")=0 AND INSTR(FSOURCE$,".com")
   =0 THEN PRINT:BEEP:PRINT "File must have .COM":END
120 FILEN=INSTR(FSOURCE$,".")-1:FILEN$=LEFT$(FSOURCE$,FILEN)
   +".BAS"
130 PRINT"Datafile [";FILEN$;"]";:INPUT ":",FDEST$
140 IF FDEST$="" THEN FDEST$=FILEN$
150 INPUT "Starting line number[100]: ", SLN
160 INPUT "Line increment [10] : ", LINC
170 INPUT "hex/decimal [h]: ", H$
180 IF SLN=0 THEN SLN=100
190 IF LINC=0 THEN LINC = 10
200 IF H$="" OR H$="h" OR H$="H" THEN H=1
210 :
220 REM Capture directory in random file
230 :
240 P1=PEEK(&H30):P2=PEEK(&H31)
250 SHELL "dir "+FSOURCE$+" > $$zztemp"
260 POKE &H30,P1:POKE &H31, P2
270 OPEN "$$zztemp" FOR INPUT AS 2
280 FOR I=1 TO 4:INPUT #2, DMY$:NEXT
290 INPUT #2, ENTRY$
300 REM Get size of com file from dir
310 SIZE = VAL(MID$(ENTRY$,16,6))
320 CLOSE #2 :KILL "$$zztemp"
330 :
340 REM Open com file and new dat file
350 :
360 OPEN FSOURCE$ AS 1 LEN = 1
370 FIELD 1,1 AS BYTE$
380 OPEN FDEST$ FOR OUTPUT AS 2
```

```
390 LINNUM = SLN+LINC
400 LIN$=STR$(SLN)+" DATA"
410 IF H=1 THEN LIN$=LIN$+" &h"+HEX$(SIZE-1) ELSE LIN$
    =LIN$+STR$(SIZE-1)
420 PRINT #2,LIN$
430 LIN$=STR$(LINNUM)+" DATA"
440 LINNUM=LINNUM+LINC
450 FOR COUNT=1 TO SIZE
460 GET #1, COUNT
470 WBYTE$=BYTE$
480 IF H=1 THEN NUM$="&h"+HEX$(ASC(WBYTE$)) ELSE NUM$
    =STR$(ASC(WBYTE$)):NUM$=RIGHT$(NUM$,LEN(NUM$)-1)
490 IF COUNT MOD 10>0 THEN 520
500 PRINT #2,LIN$+NUM$:LIN$=STR$(LINNUM)+" DATA"
510 LINNUM = LINNUM+LINC:GOTO 530
520 LIN$ = LIN$+NUM$+","
530 NEXT
540 IF COUNT MOD 10<>1 THEN LIN$=LEFT$(LIN$,LEN(LIN$)
    -1):PRINT#2,LIN$
550 CLOSE
560 PRINT:PRINT"file written":END
570 BEEP:PRINT"DOS error - aborting.":CLOSE:END
```

부분이 EXE형 프로그램의 선두에 존재하고 있기 때문이다.

그리고 출력되는 화일명을 위한 사용자 프롬프트가 나오게 되는데, 여기서 아무 입력도 없이 그냥 ↲을 치게 될 경우, 프롬프트에 나타난(사용자가 변환하고자 입력한 프로그램의 확장명을 .BAS로 바꾼) 화일명으로 출력화일이 작성되게 된다.

다음에는 DATA문으로 시작하는 문장번호와 이의 증가량을 묻는다. 아무 입력도 없이 그냥 ↲ 을 입력하게 될 경우, 그 시작되는 문장번호는 100으로 되고, 문장번호 증가량은 10이 된다. 또한 이때 원하는 시작번호와 문장번호 증가량을 입력해도 된다.

마지막으로 사용자가 입력해야 할 것은, DATA를 16진수로 할 것인가, 10진수로 할 것인가를 결정하여, 16진수는 H(Hexadecimal)를 10진수는 D(Decimal)를 치면 된다. 여기서 아무 입력없이 ↲을 칠 경우 BIN2DAT.BAS 프로그램은 DATA문에 의한 베이직 데이터를 16진수로 출력하는데, 이는 어셈블러에 의해 생성되는 .LST 화일과 비교하기 쉽게 하기 위해서이다.

그러면, 이미 BIN2DAT.BAS를 실행하기 위한 모든 데이터는 다 입력되었으므로, BIN2DAT.BAS는 입력받은 화일명으로, 실행화일을 찾아 그 화일을 베이직 프로그램의 효율 향상을 위해 베이직 데이터로 변환시킨다.

그리고 베이직 데이터를 이용한 루틴 부분을 작성한 뒤, 만든 데이터를 흡수하기 위해, MERGE"filename.ext" 명령을 사용하여 이를 이용하는 부분과 생성된 데이터를 하나로 흡수하여 만든다. 그런 다음 이를 디스크상에 저장하기 위해서는 SAVE 명령을 이용하면 된다. 이로써 BIN2DAT.BAS를 이용한 기계어 실행코드를 베이직 데이터로 변환하는 방법을 설명했다.

이제 우리가 관심을 가져야 할 부분은, 베이직 데이터로 변환하기 위해 정보를 입력받는 부분을 제외한 다른 곳에 있다.

제일 먼저 우리의 시선을 집중시키게 하는 240번, 260번 문장에 대해서 알아보자. 이는 도스 버전 2.0과 2.1을 위한 것으로, 일반적으로 우리가 현재 사용하고 있는 버전 3.XX에서는 베이직에서 SHELL 명령의 사용으로 도스명령 사용을 허용하지만, 버전2.0과 2.1에서는 이 부분에 베이직 프로그램의 시작이 놓이게 되어 베이직 프로그램이 깨지게 된다. SHELL 명령을 사용하기 전에 그 곳에 놓인 값을 대피시켜 놨다가 SHELL 명령이 끝나게 되면 원래의 상태로 복구해야 베이직의 깨짐을 막는다.

이때 SHELL 명령으로 읽은 COM형 화일의 크기로 첫번째 데이터 문을 생성한 후, 그 값을 첫번째 데이터로 저장한다. 그리고 계속되어 쓰여지는 데이터는 본고에서는 10으로 정해 한 문장에 10개의 데이터를 생성하게끔 해주고 있다. 여기서 사용자는 490번 문장의 "MOD 10"에서 이 10을 다른 수로 고칠 경우, 한 문장당 데이터의 숫자를 다르게 하여 많거나 적게 할 수 있다. 예를 들어 10대신 8을 넣으면, 한 문장당 데이터의 갯수는 8개가 된다.

이제 BN2DAT.BAS에서 중요한 부분에 대해서 알아 보았으므로, 다음에는 실제 사용되는 예를 들어 설명하겠다.

프로그램사용 예

이 프로그램은 베이직으로 프로그램을 작성할 때, 필자의 경우에 주로 사용하는 것으로 실행속도가 떨어지고, 베이직으로 구현하기 어려운 기능에 부딪쳤을 때 사용한다.

리스트 4를 보면 짐작이 가듯이 BIN2DAT를 실행시켜 특정기능을 수행하는 기계어 코드를 베이직 데이터로 변환하려면 CALL 명령어를 이용한다. 또한 리스트 3과 같이 특정 유틸리티를 제공해주는 베이직 프로그램으로도 이용할 수도 있다. 물론 체크썸을 가능하게 해주는 기능을 갖고 있지 않지만, 이 글을 읽는 독자라면 충분히 그 기능을 삽입할 수 있으리라 믿는다. 그러면 어떻게 BIN2DAT를 이용하여 예제 프로그램 리스트 3이나 리스트 4와 같이 만들었는지 알아보자.

먼저 사용자가 필요한 루틴을 어셈블리어로 작성한다. 그런 다음에 이를 어셈블하고 링크하여 EXE의 확장명을 갖는 실행화일을 얻는다. 그리고 도스의 유틸리티 프로그램의 하

나인 EXE2BIN.EXE를 실행시켜 또다른 실행화일을 COM의 확장명을 갖는 프로그램을 생성한다. 그리고 EXE 확장명을 갖는 실행화일을 지운다. 필자는 이런 일련의 작업을 배치화일로 작성하여 한꺼번에 처리하는데, 그 배치화일을 보면 리스트 5와 같다.

배치화일을 실행한 다음에 앞에서 설명한 BIN2DAT.BAS의 사용법으로 BIN2DAT.BAS를 실행시키면 된다. 끝으로 독자 여러분이 좀더 유익한 이용과 새로운 활용을 기대하며, 작은 프로그램도 이용하기에 따라 큰 기능을 발휘할 수 있다는 것을 알려드리며 이만 글을 마칠까 한다.

〔리스트 2〕 예제 프로그램

```
;
; 이 예제 프로그램을 COM 형 파일로 만든후 BIN2DAT를 실행 시켜
; 리스트 1과 리스트 2 와 같은 데이터를 얻었다.
;

code segment
     assume cs:code,ds:code

main   proc  far
       push  ax
       push  bx
       push  dx

; 한 문자씩 화면에 표시한다.

       mov   bx, offset msg
print: mov   dl, cs:[bx]
       cmp   dl, 0
       je    exit
       inc   bx
       mov   ah, 2
       int   21h
       jmp   print
exit:  pop   dx
       pop   bx
       pop   ax
       ret

msg    db 0dh, 0ah ,'이 프로그램은 BIN2DAT를'
       db '위한 예제 프로그램 입니다.'
       db 0dh, 0ah, 0

main   endp
code   ends
       end      main
```

〔리스트 3〕 특정 유틸리티를 제공해주는 예

```
20 OPEN "EXAM.COM" AS #1 LEN=1
30 FIELD #1, 1 AS A$
40 READ SIZE
50 FOR I=1 TO SIZE
60 READ BYTE
70 PUT #1
80 NEXT I :CLOSE
90 END
100 DATA &h70
110 DATA&h50,&h53,&h52,&hBB,&h19,&h0,&h2E,&h8A,&h17,&h80
120 DATA&hFA,&h0,&h74,&h7,&h43,&hB4,&h2,&hCD,&h21,&hEB
130 DATA&hF1,&h5A,&h5B,&h58,&hCB,&hD,&hA,&h20,&h20,&h20
140 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
150 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
160 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
170 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
180 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
190 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
200 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
210 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
220 DATA&hD,&hA,&h0
```

〔리스트 4〕 특정 기능을 CALL 명령으로 수행하는 예

```
10 DEF SEG=&H1700
20 READ COUNT
30 FOR MEM=0 TO COUNT
40 READ BYTE
50 POKE MEM, BYTE
60 NEXT
70 SAMPLE=0
80 CALL SAMPLE
90 END
100 DATA &h70
110 DATA&h50,&h53,&h52,&hBB,&h19,&h0,&h2E,&h8A,&h17,&h80
120 DATA&hFA,&h0,&h74,&h7,&h43,&hB4,&h2,&hCD,&h21,&hEB
130 DATA&hF1,&h5A,&h5B,&h58,&hCB,&hD,&hA,&h20,&h20,&h20
140 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
150 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
160 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
170 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
180 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
190 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
200 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
210 DATA&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20,&h20
220 DATA&hD,&hA,&h0
```

〔리스트 5〕 어셈블리 프로그램을 COM형 실행화일로 만들어 주는 배치화일과 그 실행 예

```
C:\ASSEMBLE>type assemble.bat
echo off
\masm\masm %1.asm;;;
if errorlevel 1 goto exit
\masm\link %1.obj;;;
if errorlevel 1 goto exit
del %1.bak
del %1.obj
\dos\util\exe2bin %1.exe %1.com
if errorlevel 1 goto exit
del %1.exe
:exit
echo on
C:\ASSEMBLE>assemble test↵
```

```
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1988.  All rights reserved.

 50900 Bytes symbol space free

    0 Warning Errors
    0 Serve   Errors

Microsoft (R) Overlay Linker  Version 3.64
Copyright (C) Microsoft Corp 1983-1988.  All rights reserved.

  Warning: no stack segment

C:\ASSEMBLE>
```

BBS 호스트소프트웨어(1)
와일드캣

임규남 / BBS 연구회

이번에 소개할 공개 소프트웨어는 요즘 한창 인기있는 분야인 BBS의 편리한 호스트용 소프트웨어인 와일드캣 1.0, DOS 3.1의 도움말 화면을 보여주는 PC-HELPER 그리고 두개의 디렉토리의 비교를 용이하게 해주는 DD 등이다.

와일드 캣 1.0

최근에 우리나라에서도 많은 BBS가 생기는 현상은 매우 고무적인 일이 아닐 수 없다. 1년전만 하더라도 BBS라는 말조차 생소했었던 우리나라에서 이만큼 BBS 문화가 육성되었다는 사실은 감격적인 일이다.

BBS를 운영하기 위해서는 BBS 전용의 호스트 소프트웨어가 있어야 한다. 전통적으로 성능이 좋다고 소문이 나 있는 BBS 소프트웨어로는 캐피틀 PC 유저 그룹의 RBBS-PC, Fido-BBS, TBBS, PC-보드 등이 있다. 그 외에도 Quick BBS, RYBBS, Opus 등 많이 있다. 하지만 공개소프트웨어로 나와 있는 것은 몇개되지 않는다.

공개소프트웨어나 쉐어웨어로서 제법 괜찮은 프로그램을 찾는다면 단연 RBBS-PC와 와일드캣을 들 수 있다. (물론 그 외에도 많이 있다). 이 두 프로그램은 우리나라에도 제법 많이 퍼져있기 때문에 구하기가 매우 쉬울 것이다. 그래서 이번달에는 와일드캣에 대해서 설명하고 다음달에는 RBBS-PC에 관해서 설명하겠다. 다른 BBS 소프트웨어의 경우에도 자료가 정리되는 대로 계속 소개할 예정이다.

(1) 인스톨

와일드캣은 현재 버전2.0까지 나와 있지만 이 버전은 상업용으로 쓰이고 있는 실정이다. 그리고 버전1.0의 실행 가능한 최소 메모리 용량이 320KB 정도여서 여기서는 버전1.0을 소개하기로 한다.

와일드캣 1.0은 플로피 디스크 2장에 들어 있다. 우선 하드디스크에 와일드캣 전용의 서브디렉토리를 하나 만들어야 한다. 앞으로의 예에서는 이 서브디렉토리의 이름을 "WILDCAT"이라고 하자. 그러면 이제 DOS상에서 와일드캣 디스켓을 A드라이브에 넣은 후 다음과 같은 명령을 내려서 와일드캣을 하드디스크에 복사한다.

```
C:\>MD WILDCAT
C:\>CD WILDCAT
C:\WILDCAT>COPY A:*.*
```

위의 마지막 카피 명령을 2번 디스켓을 드라이브 A:에 넣고 반복한다. 그러면 와일드캣의 모든 화일이 하드디스크의 WILDCAT이라는 서브디렉토리에 복사가 된 것이다. 다음으로 할 일은 CONFIG.SYS 화일의 내용을 다음과 같이 만든다.

```
FILES=20
BUFFERS=40 (DOS 3.x 일때)
```

DOS 2.X를 사용하는 유저는 위의 버퍼의 크기를 20으로 하기 바란다. 이렇게 했으면 MAKEWILD라는 셋업 프로그램을 실행시킨다.

(2) 셋업 프로그램 실행

이제 WILDCAT 서브디렉토리로 가서 MAKEWILD를 실행시킨다. MAKEWILD 는 와일드캣의 각종 파라미터를 고칠 때 사용하는 셋업 프로그램이다. 셋업 프로그램을 실행시키면 이 프로그램은 우선 CONFIGWC.BBS와 MENUINFO.BBS라는 두개의 화일을 찾는다. 이 두개의 화일안에는 와일드캣의 각종 파라미터가 저장되어 있는데 처음 셋업을 하게 되면 이 화일이 없기 때문에 셋업 프로그램이 이 두 화일을 디폴트값으로 생성한다.

셋업 프로그램을 실행시킨 직후의 화면은 화면 1과 같다. 화면의 왼쪽부분이 각 BBS기능의 나열이고 오른쪽의 숫자는 각 기능의 셋업 항목이 있는 페이지(또는 프레임) 번호이다.

우선 처음으로 셋업 프로그램을 실행시켰으면 일단 '0'을 입력하고 〈F1〉키를 누른다. 이렇게 하면 디폴트값으로 MENUINFO.BBS, CONFIGWC.BBS라는 두 개의 화일이 생성되는 것이다.

다시 셋업 프로그램을 실행시켜서 화면 1로 돌아온다. 이때 화면 1과 같은 각 화면을 프레임이라고 부른다.

각 프레임의 항목을 수정하는데 사용되는 키는 다음과 같다.

←: 현재 필드에서 왼쪽으로 한글자 이동

→: 현재 필드에서 오른쪽으로 한글자 이동

〔화면 1〕 셋업 프로그램(MAKEWILD) 실행화면

```
            WILDCAT (tm) INSTALLATION PROGRAM            Screen 1

                  Version 1.00  (03/22/87)

                  Section                          Screen

        General Information about the Board & Sysop   2
        Modem Information & Settings                  3
        New User Defaults                             4
        User Defaults                                 5
        Security Information                          6
        System Configuration                          7
        Message & File Folder                      8..10
        Define your Dynamic (Smart) Menu's        11..14
        Security Level Definitions                15..16
        Define for File Paths                        17
        Quit & Save This Installation                 0

          Which Page would you like to go to ? 2

  Select a Starting Page and Press F1
                                              Insert Off
```

↑, ↓ : 위나 아래의 필드로 이동
↲(ENTER키) : 다음 필드로 이동
Home : 현재 필드의 첫번째 컬럼으로 이동
End : 현재 컬럼의 마지막컬럼으로 이동
PgUp : 전 프레임으로 이동
PgDn : 다음 프레임으로 이동
Ins : 삽입 모드로 토글
BackSpace : 왼쪽 컬럼의 글자 지우기
Del : 커서밑의 글자 지우기

각 프레임의 항목중에서 중요한 부분만 설명하겠다. 일단 '2'를 입력하고 〈F1〉키를 누르면 프레임 2로 가게 된다. 각 프레임간의 이동은 위에서도 설명하겠지만 〈PgUp〉, 〈PgDn〉키를 이용하면 된다. 우선 프레임 2에서 중요한 것은 시삽의 이름이다. (항목 1, 2) 시삽(SYStem OPerator)이라는 것은 각 BBS의 운영을 맡는 사람을 말하는 것이다. 다음으로 BBS의 이름이다.(항목 3)

뒤에서도 마찬가지이지만 시스템에 그렇게 중요하지 않은 항목은 설명을 생략하겠다.

다음 프레임 3으로 가게 되면 모뎀에 관한 항목이 있다. 우선 COM 포트를 정해주고(항목 1), 캐리어 검출까지의 시간을 정하고 (항목 2), 모뎀의 리셋명령과 초기화명령을 입력한다. (항목 3, 4) 가장 중요한 것은 항목 5 의 모뎀 속도이다. 현재 1200BPS (Bit Per Second)가 우리나라 BBS의 주류를 이루고 있지만 점차 2400BPS가 널리 퍼질 것이다.

다음 프레임 4로 가게 되면 새로 가입하는 유저들의 등록 여부를 결정하는 항목이 있다. 우선 항목 1은 BBS를 폐쇄적으로 운영하려는지의 여부를 묻고 항목 2는 새로 가입하는 유저에게 주어지는 시큐리티 레벨을 정한다. 시큐리티 레벨이라 하면 각 유저에게 할당하는 등급 비슷한 것으로서 일반적으로 시큐리티 레벨이 높을수록 BBS 사용에 제한이 없다. 다르게 말하면 시큐리티 레벨이 너무 낮으면 BBS의 메뉴 사용에 상당한 불편을 느끼게 된다. 항목 3, 4, 5는 각각 새로운 유저에게 전화번호, 생일, 컴퓨터 기종을 물을지에 대한 것이다. 이 항목들을 세트시켜 놓으면 새로 가입하는 유저는 자신의 전화번호, 생일, 컴퓨터 기종 등을 남겨야만 한다. 항목 7은 처음 호출했을 때의 사용시간을 정한다. 일반적으로 처음 호출하는 유저는 서투른 경우가 대부분이므로 시간을 평상시보다 많이 주는 경향이 있다.

다음 프레임 5로 가게 되면 평상시의 호출에서의 시간 항목 1과 하루에 다운로드 가능한 최대의 바이트수를 입력하게 된다(항목 2).

프레임 6으로 가면 BBS의 시큐리티(안전)에 관한 항목들이 있다. 항목 2, 3은 매번 몇번의 호출마다 전화번호와 생일을 확인함으로써 각 유저의 암호 누출을 방지하는 기능이다. 또 항목 8은 유저의 레벨의 갯수를 결정하는 것이다. 예를 들어서 BBS 안에 1, 10, 100, 1000의 4개의 레벨을 만든다면 항목 8에 '4'를 입력하면 된다.

프레임 7에서는 시스템에 대한 항목이 있다. 항목 3은 유저가 업로드를 했을 때 시간보상 비율을 입력한다. 예를 들어서 여기서 3을 입력하면 유저가 10분짜리 공개소프트웨어를 업로드 했을 때 30분의 여유시간을 돌려주게 되는 것이다. 항목 5는 업로드 대 다운로드의 비율의 경고 수준을 말한다. 예를 들어 항목 5에 '10'을 입력하면 유저의 업로드대 다운로드 비가 1 : 10을 넘었을 때 경고 메시지가 나가면서 유저 데이터 화일에 기록되게 된다. 항목 7은 현재 게시되어 있는 게시판 화일의 갯수를 말한다.

프레임 8에서는 메일과 화일 영역에 대한 정보를 기록하게 된다. 항목 1은 메일 폴더의 갯수 항목이다. 메일 폴더라 하면 보통 컨퍼런스라고도 알려진 것으로서 특정 주제에 대한 메일 코너를 의미한다. 항목 2는 화일영역의 갯수 항목이다. 화일 영역은 공개소프트웨어를 각 분야별로 정리해서 보관하는 영역을 말한다. (실제로 프레임 17에서 각 영역의 서브디렉토리를 지정하게 된다.)

프레임 9에서는 각 메일 폴더의 이름을 입력해 주게 되고, 프레임 10에서

는 각 화일 영역의 이름을 입력하게 된다.

프레임 11은 메인 메뉴의 각각의 명령에 대한 시큐리티 할당과 화면에 출력되는 순서 등을 입력하고, 프레임 12에서는 메시지 메뉴에 관한 것을, 프레임 13에서는 화일 메뉴에 관한 것을, 프레임 14에서는 시삽 메뉴에 관한 것을 입력하게 된다. 프레임 11, 12, 13, 14를 보게 되면 〈Activity〉, 〈Call Ltr〉, 〈Description〉, 〈Seq〉, 〈Sec〉 등 5개의 설명이 맨 윗줄에 있다. 각 항을 소개하면 우선 〈Activity〉는 실제 와일드캣에서 일어나는 일을 묘사하는 것으로서 BBS 시삽이 수정할 수 없는 하드 코드화된 부분이다. 〈Call Ltr〉는 메뉴에서 명령을 사용할 때 이용하는 문서이다.

보통 뒤의 〈Description〉부분 첫글자와 같지만 반드시 같지 않아도 된다. 〈Description〉은 실제 와일드캣이 실행되었을 때 화면에 나가는 문장이다. 〈Seq〉는 앞의 〈Description〉이 화면의 메뉴항에 나가는 순서를 결정하고 코맨드라인의 글자 순서도 결정하게 된다. 〈Security〉는 이 수치이상의 레벨을 가진 유저만이 해당 항목을 볼 수 있게 하는 것이다. 즉, 특정 메뉴를 일반 유저에게 제공해주지 않고 싶으면 그 메뉴의 시큐리티를 높게 잡고 특정 유저의 시큐리티 레벨만을 높게 주면 되는 것이다. (특히 시삽 메뉴같은 것)

프레임 15로 가게 되면 시큐리티 레벨에 관한 것을 입력한다. 맨 왼쪽의 〈Sec Lev〉가 시큐리티 레벨의 수치를 입력받는 것이다. 이 수치는 시삽이 임의로 정하는 것이다. 예를 들어서 1, 2, 3,… 이렇게 나가도 되고, 1, 10, 100, 1000… 이런 식으로 나가도 되는 것이다. 두 번째의 〈Allow in these Message Folders〉 항목은 접근 가능한 메일 폴더를 제한하는 것이다. 여기에서 사용되는 메일 폴더의 이름은 프레임 9에서 입력한 것을 사용한다. 〈Allow to Download in these File Area〉 항목은 다운로드가 가능한 영역을 지정한다. 〈DTL〉은 Daily Time Limit로서 하루에 사용할 수 있는 최대시간을 말한다.

프레임 16으로 가게 되면 각 시큐리티 레벨별로의 업로드 가능한 화일 영역과 하루에 다운로드 가능한 최대 화일수를 지정하게 된다(Maximum number of Downloads allowed per day 항목).

프레임 17은 프레임 10에서 정한 화일 영역의 이름에 대한 실제 디스크상의 서브디렉토리를 지정해 주게 된다.

여기까지 소개한 항목은 꼭 여러분 자신이 손을 봐야(?)할 중요한 부분들이다. 프레임 17에서 〈PgDn〉키를 누르면 다시 화면 1로 가게 된다. 여기서 '0'을 입력하고 〈F1〉키를 누르면 위에서 입력한 내용들이 CONFIGWC.BBS와 MENUINFO.BBS에 기록된다. 나중에 다시 이 항목에 대해서 수정할 일이 생기면 다시 셋업 프로그램을 실행시켜서 해당 항목만을 수정하고 나오면 된다. 이제 셋업이 끝났으니 직접 BBS에 들어가 보기로 하자.

(3) 와일드캣

와일드캣(WILDCAT)을 실행시키면 무스탕 회사의 로고가 나오고 와일드캣의 상황판(?)이 나올 것이다. 모든 것이 제대로 셋업되었다면 잠시후 호출에 대한 준비가 끝났다는 메시지가 나온다(화면 2 참조). 여기서 〈Home〉키를 누르면 이 상황판에서 사용할 수 있는 키 명령에 대한 도움말이 나온다(화면 3 참조).

여기서 사용되어지는 키는 유저가 로그온 되어있을 때에도 적용이 되는 명령이므로 잠시 설명을 하고 넘어가자.

〈F1〉: 시삽이 직접 로컬로 로그온할 때

〔화면 2〕 호출에 대한 준비 메시지 화면

```
   WILDCAT! Version 1.00  (03/22/87)        Registration Number: 00000
                              Devirus BBS
      No calls have been received since the system was initialized.

   Database Information:                    Toggles:
        Users        = 0                         Printer    is OFF
        Files        = 0                         Pager      is OFF
        Messages     = 0                         Page Bell  is OFF
        Total Calls  = 0                         Keyboard   is OFF

   Available Disk Space:
        Drive C has    4,687,872

  WILDCAT! ready for calls at 10:15 on Saturday, the 10th of June 1989.

             This SHAREWARE version may be freely distributed.
           Press [HOME] for description of command menu options.
 WILDCAT!  (C) 1986, 87 By Mustang Software, Bakersfield, CA  (805) 395-0223
```

〔화면 3〕 도움말 화면

```
   WILDCAT! Version 1.00  (03/22/87)       Registration Number: 00000
 ➡ Sysop Help Menu ⬅
            F1 - Log-on Sysop Locally            F2 - Not Used
            F3 - Toggle the printer              F4 - Toggle Sysop Page
            F5 - Toggle Page Bell                F6 - Toggle Screen Writting
            F7 - Begin Chat/ Answer Page         F8 - End Chat
            F9 - Upgrade User ONLINE            F10 - Terminate WILDCAT!

 ALT+L - Normal Log-on        Arrow-Up   - Add 5 minutes to Time
 ALT+D - Dos Shell            Arrow-Dn   - Reduce 5 Minutes from Time
 ALT+K - Keyboard on/off      ALT+ 1..4  - Send "ALTx.BBS" & Log-off
 ALT+W - Sysop Window Toggle  ALT+ 5..9  - Send "ALTx.BBS" only
 Home  - This window          ALT+0      - Send "ALT0.BBS", Log-Off, & Log-Out

                WILDCAT! Bulletin Board System version 1.00

             This SHAREWARE version may be freely distributed.
           Press [HOME] for description of command menu options.
 WILDCAT!  (C) 1986, 87 By Mustang Software, Bakersfield, CA  (805) 395-0223
```

사용하는 키이다.

〈F6〉: 화면의 온/오프 여부를 결정한다. 이것이 오프되어 있으면 유저의 행동이나 키입력이 화면에 보이지 않는다. 처음 BBS를 개설한 유저라면 이것을 온시켜놓는 것이 좋다.

〈F7〉: 체팅시작. 유저에게 시삽이 체팅을 걸거나 유저의 시삽 호출(페이지 시삽이라고 한다.)에 대해 답할 때 사용한다.

〈F8〉: 유저와의 체팅을 끝낼 때 사용.

〈F9〉: 현재 로그온되어 있는 유저의 레벨을 올려줄 때 사용한다. 이 키를 누르면 수치를 입력하는 윈도우가 나타나는데 여기에서 원하는 시큐리티 레벨의 수치를 입력하면 현재의 로그온 된 유저의 레벨이 그 수치로 된다.

〈F10〉: 와일드캣을 끝낼 때 사용한다.

〈ALT-K〉: 키보드 모드의 온/오프를 결정한다. 이것이 온되어 있으면 시삽이 입력하는 것이 마치 로그온된 유저가 입력하는 것으로 인식한다. 이 기능을 사용할 때는 특히 처음 로그온 하는 유저가 난처한 상황을 대했을 때 시삽이 이의 등록을 도와주고자 할 때이다.

〈ALT-L〉: 로컬 로그온을 한다. 〈F1〉과 비슷하지만 〈F1〉은 시삽으로서 로그온이 되는 것이고 〈ALT-L〉로 하게 되면 일반 유저와 같은 방법으로 로그온이 되기 때문에 시삽외의 유저도 테스트가 가능하다.

여기서 아래, 위 화살표키가 상당히 중요한 키이다. 위화살표키는 로그온된 유저의 시간을 5분단위로 늘려줄 때 사용하고 아래 화살표는 로그온된 유저의 시간을 5분단위로 줄여줄 때(?) 사용한다.

이제 〈F1〉키를 눌러서 로컬로 들어가자.

와일드캣이 묻는 질문에 대답을 하고서 메인메뉴로 들어오면 〈F9〉키를 눌러서 최고 레벨을 자신(시삽)에게 할당한다.

이제는 모든 메뉴를 사용할 수가 있다. 각 메뉴의 기능을 연습 해보기 바란다. 유저관리나 화일 관리는 메인 메뉴에서 〈R〉키를 사용해서 시삽메뉴로 가게 되면 다 들어 있으므로 그것을 활용하기 바란다.

(4) 와일드캣의 텍스트 화일

와일드캣을 본격적으로 운영하기 위해서는 몇몇 텍스트화일이 필요하다. 다음에 설명할 화일들을 일반 에디터(사이드킥의 노트패드나 워드스타의 논도큐먼트 옵션)를 사용하여 작성하

〔화면 4〕 PRINT.COM의 도움말

```
01/01/80                                                  00:01:13

PRINT - allows the queueing of text files to a printer.  External
        command.
Syntax:  PRINT [[/D:device][/B:bufsiz][/U:busytick][/M:maxtick]
        [/S:timeslice][/Q:quesiz][/T][/C][/P]  [d:]filename[.ext]...]
where /B:bufsiz - set # of bytes of internal buffer (default 512KB)
/Q:quesiz - set # of files in the queue (range is 1-32, default is 10)
/S:timeslice - set timeslice value (default is 8, range is 1-255)
/U:busytick - set # of clockticks to wait for printer to become available
             (default is 1)
/M:maxtick - set # of clockticks to print characters on the printer (default
             is ?, range is 1-255)
/T = cancels files in the queue    /D:device = specifies print device
/P = add files to the queue     /C = permits selected files to be cancelled

                    PRESS 'F1' TO RETURN TO HELP SCREEN
                    PRESS 'F10' TO EXIT PROGRAM
```

〔화면 5〕 DD의 화면

```
DoubleDir ♦ (c) 1985 Eric Grunin                        F1 Help ♦ Esc to exit
________________________________________  ________________________________________
.               <DIR>    1-01-80  12:10a ABCBLOCK.LXH    3382   7-09-85   3:56p
..              <DIR>    1-01-80  12:10a ABCBLOCK.LXM    1822  11-28-84   2:41p
4201.CPI        17089    3-18-87  12:00p ARTDECO.LXH     4942   6-23-85   5:30p
5202.CPI          459    3-17-87  12:00p ARTDECO.LXM     2662   3-02-85   1:15p
A                2560    1-01-80   7:06a BANKER.LXH      4994   6-27-85   3:23p
ANSI.SYS         1678    3-17-87  12:00p BANKER.LXM      2690   3-10-85   7:29p
APPEND.EXE       5825    3-17-87  12:00p BRO.COM         8768   1-09-86   8:36p
ASSIGN.COM       1561    3-17-87  12:00p BROADWAY.LXH    4942   8-27-84   1:07p
ATTRIB.EXE       9529    3-17-87  12:00p BROADWAY.LXM    2662   1-13-85   3:39p
BACKUP.COM      31913    3-18-87  12:00p BROCHURE.BAT     383   1-08-86   8:28p
BASIC.COM        1063    3-17-87  12:00p COURIER.LXH    13210  12-09-85   4:30p
BASIC.PIF         369    3-17-87  12:00p COURIER.LXM     7114   1-01-80  12:06a
BASICA.COM      36403    3-17-87  12:00p CYRILLIC.LXH    4942   7-13-85   8:43a
BASICA.PIF        369    3-17-87  12:00p CYRILLIC.LXM    2662   7-11-85   6:01p
C                 128    1-01-80   7:48a DEALER            71   1-01-80  12:00a
CHKDSK.COM       9850    3-18-87  12:00p DEALR             55   1-01-80  12:00a
COMMAND.COM     25307    3-17-87  12:00p DEMO.BAT         344   1-08-86   8:28p
COMP.COM         4214    3-17-87  12:00p DEMO.LXT       25402   1-08-86   8:08p
COUNTRY.SYS     11285    3-17-87  12:00p ENGRAVED.LXH    4942   1-21-85  12:27a
DEBUG.COM       15897    3-17-87  12:00p ENGRAVED.LXM    2690   7-27-84   5:25p
DISKCOMP.COM     5879    3-17-87  12:00p EPSON.PRN        137  12-15-85   6:16p
c:\dos\*.*                               a:\*.*
```

기 바란다.

BULLETIN.BBS : 게시판의 메뉴 화일

BULLETxx.BBS : 각 게시판 화일. 여기서 xx는 0에서 99까지의 숫자이다. 예를 들어서 BULLET17.BBS는 게시판 화일 17번이다.

GOODBYE.BBS : 유저가 로그오프할 때 나가는 인사메시지가 들어있는 화일이다.

HELLO1.BBS : 유저가 로그온 했을 때 나가는 환영 메시지가 들어있는 화일이다.

이 외에도 몇가지가 더 있지만 그리 중요한 것은 아니다. 더 알고 싶은 독자는 와일드캣 디스켓에 있는 도큐먼트 화일을 프린트해서 42쪽에서 46쪽까지 참고하기 바란다.

PCHELP

PC-DOS 3.1의 도움말을 보여주는 프로그램이다. 실제로 필자 자신은 계속해서 PC-DOS 3.1만을 사용하고 있기 때문에 상당히 유용했다. 각 DOS 외부 명령(예를 들어서 PRINT.COM)에 대한 스위치와 설명이 잘 나와 있다.

PCHELP.COM을 유틸리티나 DOS 서브 디렉토리에 넣어놓고 패스를 지정해서 사용하면 된다.

화면 4는 PRINT.COM의 도움말을 보는 예이다.

DD(Double Dir)

두 개의 디렉토리를 동시에 보여주는 유틸리티이다. 양옆으로 각각 디렉토리를 보여주는 데 각 디렉토리는 독립적으로 스크롤이 된다(화면 5 참조).

DD에서 사용되는 키는 다음과 같다.

〈PgDn〉, 〈F8〉 : 앞으로 한 페이지 이동
〈PgUp〉, 〈F7〉 : 뒤로 한 페이지 이동
위 화살표 : 앞으로 한 줄 이동
아래 화살표 : 뒤로 한 줄 이동
Home : 디렉토리의 처음으로
End : 디렉토리의 끝으로
〈F1〉 : 도움말
〈F9〉 : 프린트
〈Esc〉 : DOS로 빠져나감 PC

□ 특집

CP/M-86의 분석

CP/M-86은 i8086 CPU에서 사용 가능하도록 만들어진 OS로서 타기종으로의 이식성이 유닉스 다음으로 높다는 평가를 받고 있다. 본고에서는 CP/M을 구성하고 있는 BDOS, BIOS, CCP, TPA를 중심으로 CP/M-86의 모체인 CP/M-80과의 차이점과 내부구조 등 CP/M-86의 전반적인 이해와 함께 시스템콜에 대해서도 알아보고자 한다.

김종석/챌린즈 기술부



1970년대의 APPLE을 중심으로 한 마이컴의 춘추 전국시대는 1980년대에 이르러 푸른색 거인(Big Blue)이라 불리는 IBM의 무차별 공격으로 전세계가 IBM-PC로 평정된 듯한 느낌이다. 그러나 그후에는 주된 소프트웨어의 기준이 되고자 하는 경쟁이 또한 전쟁을 방불케하고 있다. 즉, 8비트 PC 이후 지속적으로 왕자의 자리를 지켜온 CP/M과 멀티유저/멀티태스킹 및 강력한 이식성을 내세운 UNIX 그리고 소프트웨어의 대부 마이크로 소프트사의 MS-DOS 등이 새로운 PC의 주인이 되고자 싸움을 벌여왔다. 현재에 이르러서 MS-DOS와 UNIX계열이 이 PC를 양분하고 있다고 생각된다.

그러나 CP/M은 지금까지 쌓아온 역량으로 보아 앞으로도 계속 살아 남을 OS인 것 또한 사실이다. 그것은 아직도 CP/M으로 작성된 많은 응용프로그램이 있고 이를 즐겨 사용하는 사용자가 많기 때문이다. 이제 CP/M-80에서 CP/M-86으로 전환하려는 독자와 CP/M에 관심이 많은 MS-DOS사용자 그리고, CP/M-80용 응용 프로그램을 계속해서 사용하려는 독자를 위해 기본개념부터 그 내부구조에 이르기까지 되도록 쉽게 설명하고자 한다. 단 본고는 독자가 적어도 CP/M-80의 기본적인 명령어를 알고 있다고 생각하고 서술할 예정이다. 만일 CP/M의 명령어를 알고자 한다면 기존에 나온 CP/M에 대한 서적을 참조바란다.

CP/M이란?

(1) OS란?

만일 우리가 미국에 가려고 한다면, 먼저 미국으로부터 입국허가서를 받아야 한다. 즉 비자(Visa)가 필요하다. 이 비자를 받게되면, 미국의 여러 도시를 여행할 수 있게 되는 것이다. 마찬가지로, 우리가 어떤 컴퓨터에서 특정한 응용 프로그램을 돌리고자 한다면, 거기에는 기본적으로 OS가 필요하다. 이 OS를 기본으로 하여 모든 응용 프로그램이 수행되는 것이다.

그런데, 만일 미국 비자를 갖고 프랑스의 파리나 영국의 런던을 방문하고자 한다면, 그것은 불가능한 일이 될 것이다. 이와 같이 OS도 그 OS에 맞는 응용 프로그램만을 수행시킬 수 있다. 즉, OS란 컴퓨터와 인간 그리고 응용 프로그램을 자연스럽게 연결시켜 원활한 대화 및 작업이 가능하도록 해주는 기본적인 시스템 프로그램인 것이다.

따라서 OS에는 그 종류도 다양하고 사용방법 또한 상이하기 때문에 사용자 및 프로그램 설계자에게 얼마만큼 쉽고, 편하느냐에 따라 그 OS의 성패가 달려 있다고 말할 수 있을 것이다.

(2) CP/M의 역사

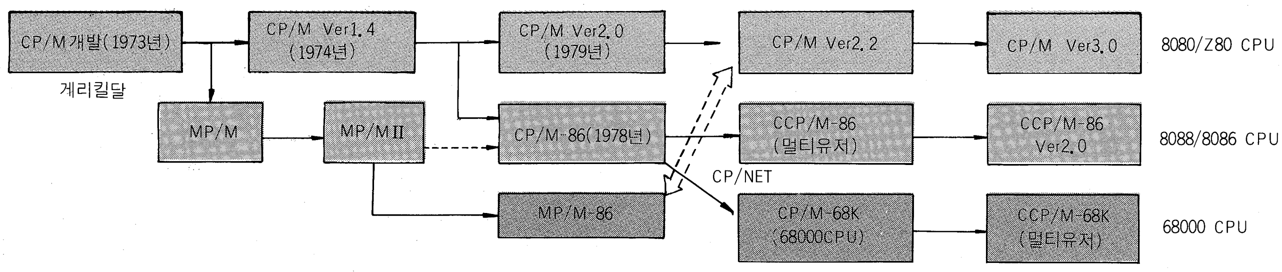
CP/M은 1973년, 디지탈 리서치사(Digital Research Inc.)의 게리 킬달(Gary A. Kildal)에 의해 개인용 컴퓨터를 위한 OS(Control Program for MicroComputer)로서는 최초로 개발되었다. 킬달은 인텔사에 있을 때, PL/M(Programming Language for MicroComputer) 컴파일러를 제작하였다. 이것은 인텔사의 소프트웨어 시스템 구현언어가 되었으며 그는 곧 이 PL/M 컴파일러를 지원하기 위한 램상주 프로그램을 작성하였는데 그것이 바로 CP/M의 초기 버전이 되었다.

이 당시, 킬달은 플로피디스크 시스템과의 인터페이스에 역점을 두고 CP/M을 작성하였는데, 이러한 시도가 CP/M을 성공으로 직결시켰다. 이것은 당시 보조 기억장치로 사용된 종이 테이프가 가격 및 성능에서 비교가 되지 않았기 때문이다. 또한 킬달의 데이터 구조를 중시한 설계사상(Philosophy)이 CP/M에 그대로 반영되어 CP/M은 높은 이식성과 호환성을 갖는 OS가 되었다. 이에 따라 이 OS는 곧 타 마이크로 컴퓨터 제작회사에서 수용하여 급속히 전파되어 갔다.

CP/M의 한 줄기로 또 하나의 시스템은 MP/M(Multiprogramming CP/M)이다. 이 시스템은 동시에 여러명이 작업을 할 수 있게 함으로써 8비트 컴퓨터를 저급의 미니컴퓨터와 경쟁할 수 있게 했다. 1980년에 등장한 CP/NET는 CP/M을 기반으로 한 사용자를 MP/M을 서버(Server)로 하는 시스템과 네트워크 구성을 가능하게 하였다.

한편, 반도체 기술의 급격한 발전은 8080/Z80 프로세서를 중심으로 하는 컴퓨터시대에서 8086/80286, 68000 등의 프로세서를 중심으로 하는 컴퓨터로 변화

〔그림 1〕 CP/M의 발전



해 갔고, 이에 따라 등장한 것이 CP / M-86과 CP / M-68K이다(그림 1 참조).

(3) CP / M의 구성

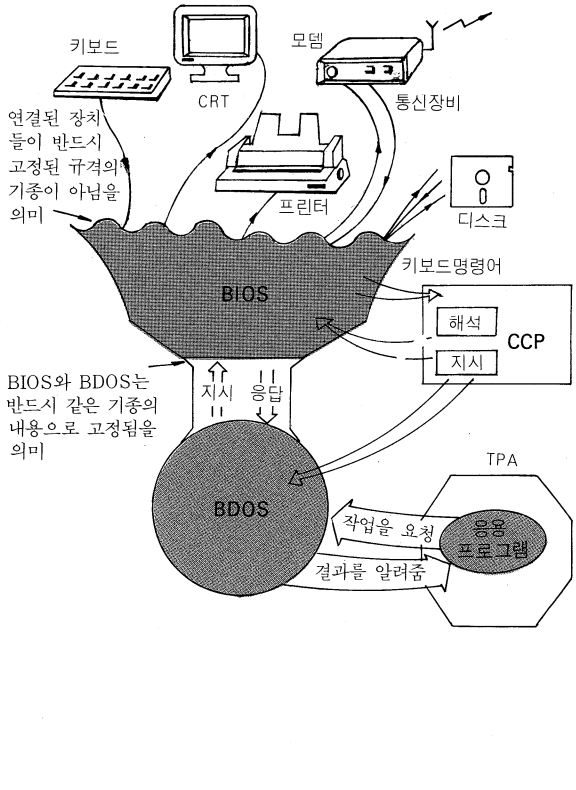
CP / M은 BIOS, BDOS, CCP, TPA 등의 4가지 부분으로 구성되어 있다. CCP(Console Command Processor) 는 사용자가 콘솔(키보드+CRT)로 입력한 명령을 해석하고 응답하는 프로세서이다. 이것은 MS-DOS의 COMMAND.COM에 해당하는 부분으로 COMMAND.COM처럼 화일로 되어있는 것이 아니라 하나의 시스템에 한 모듈로 존재한다. 이 CCP는 BDOS와 BIOS를 통해 실질적인 입 / 출력을 행한다.

BDOS(Basic Disk Operating System)는 디스크 관리를 행하는 여러 모듈로 구성되어 있으며, CP / M의 핵심을 이룬다. BDOS는 BIOS와 그 점프벡터를 통해 인터페이스되므로 호환성을 유지하는 주요부분이 된다. 즉, 전혀 다른 컴퓨터시스템에서도 변경없이 사용될 수 있는 부분이다.

BIOS(Basic Input / Output System)는 직접적으로 주변장치와 연결되는 부분으로 하드웨어에 종속된 부분이다. 따라서 하드웨어가 달라진다면, 이 부분을 패치(Patch)하여야 한다. 그러나 BDOS와 연결되는 점프벡터는 그대로 유지되어야 한다. 여기에서 말하는 BIOS는 IBM-PC의 ROM-BIOS를 의미하는 것이 아니라 시스템 디스켓에서 제공되는 CP / M BIOS를 의미하는 것에 유념하길 바란다. CP / M-86에서는 이 BIOS가 ROM-BIOS와 인터페이스되어 있다.

TPA(Transient Program Area)는 실제 응용 프로그램이 사용할 수 있는 부분으로 위에서 언급한 세 부분을 제외한 메모리 영역이다(그림 2 참조).

〔그림 2〕 CP/M 각 모듈의 상호관계



(4) CP / M-86의 내부명령어

CP / M의 내부명령어(Built-in Command)는 CCP 모듈안에 내장된 형태의 명령어를 의미하며, 이것은 특별한 화일이 존재하지 않아도 단지 시스템만 로드되어 있다면 수행가능한 명령어이다. CP / M-86에서 제공되는 내부 명령어는 DIR, DIRS, TYPE, ERA, REN, USER 등 6개이며 이들의 용도는 표 1에 나타내었다.

여기서 주의할 사항은 MS-DOS에서는 어떤 대치명령에 대해 소스(Source)가 먼저 오고 다음에 그 타겟(Target)이 오는데 반해, CP / M에서는 그 반대로 되어 있다는 점이다. 만일 명령어 모드(Command Line)에서 한화일의 이름을 다른 것으로 바꾸고자 할 때, MS-DOS에서는

```
A>RENAME A.ASM B.ASM
```

과 같이 표현하나, CP / M에서는

```
A>REN  B.ASM=A.ASM
```

으로 표현함으로써 차이점을 보여 주고 있다.

이러한 표현의 차이는 운영체계의 설계자들이 다른 삶들의 표현 방법을 그대로 사용하지 않으려는 것과 동시에 특허 및 저작권 등의 문제가 제기되는 현실을 감안한 때문이라는게 필자의 견해이다. 이 때문에, 필자가 처음 MS-DOS상에서 사용할 당시는 번번히 이런 오류를 범하여, 나중에는 CP / M상에서는 DOS 명령어의 표현방법을 사용하고, DOS상에서는 CP / M의 것을 사용하는 웃지 못할 촌극을 벌이기도 했다.

〔표 1〕 CP/M-86의 내부명령어

내부명령어	용　　도
DIR	디스크에 등록되어 있는 화일명의 리스트를 표시하는 명령어
DIRS	.SYS 속성이 붙는 화일을 표시하는 명령어 (CP/M-86에서 추가되었음.)
TYPE	지정한 화일의 내용을 표시하는 명령어
ERA	지정된 화일을 지우는 명령어
REN	화일명을 새로운 화일로 바꾸는 명령어
USER	사용자 영역을 바꾸는 명령어로 하드디스크와 같은 대용량의 보조기억장치를 여러 사용자가 분할하여 사용할 수 있도록 하는 것임.

(5) CP/M-86의 외부 명령어 (Transient Command)

CP/M-86용 시스템 마스터를 구입하게 되면 화면 1과 같은 프로그램이 따라오게 되는데, 이 프로그램들은 시스템을 사용하는 기본 명령어들로서 반드시 그 사용법에 대해 충분한 연습이 이루어져야 한다. 본고에서는 이글을 읽는 독자 여러분이 이들 명령어의 형식에 대해서는 어느정도 알고 있다고 생각하고 표 2에서 그 기능만을 간단히 나타내었다.

그리고 CP/M을 사용해본 적이 없는 MS-DOS의 독자를 위해 각 명령어에 대응하는 MS-DOS의 명령어를 같이 나타내었다.

CP/M-86과 CP/M-80

(1) CP/M-80과 상이한 CP/M-86의 명령어

CP/M-86의 명령어는 기본적으로 CP/M-80과 차이가 없다. 차이점이라면 SAVE 명령어가 없어지고, DIRS 명령어가 추가된 점이다. DIRS는 .SYS 속성을 가진 화일을 보여주는 명령어이다. 또 시스템 디스켓에 HELP.CMD가 추가되어 각 명령어에 대한 도움말을 주므로, CP/M-80 사용자나 MS-DOS 사용자가 쉽게 그 명령어를 이용할 수 있게 되어 있다. 따라서, 기존의 CP/M 머신을 사용하던 독자는 거의 아무런 불편 없이 CP/M-86과 대화할 수 있을 것이다(표 3 참조).

(2) CP/M-80 시스템 수준에서의 호환성

그러면 과연 CP/M-80에 사용되는 프로그램을 CP/M-86에서도 그대로 사용할 수 있는가라는 문제를 짚고 넘어가야겠다. 이 말은 같은 OS라면 모든 프로그램이 동일하게 수행된다고 하는 원칙(호환성)을 검증해야 한다는 말이다.

물론 이것은 CP/M-86에서도 적용되지만, 실제로 그 대답은 "어셈블리 수준에서는 아니오"이다. 그 이유는 무엇인가? 그것은 바로 그 베이스를 이루고 있는 CPU의 차이 때문이다. 그러나, 같은 언어를 사용하는 원시 프로그램을 CP/M-86의 환경하에서 재컴파일한다면, 그 호환성은 그대로 유지된다. 그러면, 어셈블리 수준에서는 전혀 호환성이 없는 것일까?

그렇지 않다. 어셈블리어에서는 시스템 수준에서 호환성을 유지하고 있다. 즉, CPU의 상이함으로 인한 기계어 수준의 호환성은 없을지라도 대응하는 레지스터와 시스템 콜(System Call)의 적절한 변환을 통해서, 8080/Z80 CPU상에서 작성한 어셈블리 프로그램이 운영체계의 시스템 콜을 사용하여 주변기기와 입/출

〔화면 1〕 CPM-86용 시스템 마스터에 들어 있는 프로그램 목록

```
A>
A>dir
A: ASM86    CMD : TOD      CMD : COPYDISK CMD : DDT86    CMD
A: ASSIGN   CMD : NEWDISK  CMD : FUNCTION CMD : PROTOCOL CMD
A: ED       CMD : SPEED    CMD : PIP      CMD : SUBMIT   CMD
A: HELP     CMD : HELP     HLP : STAT     CMD : GENCMD   CMD
A: DUMP     A86 : DUMP     LST : DUMP     H86 : DUMP     SYM
A: F1       $$$ : DUMPDISK LST : DUMPDISK H86 : DUMPDISK SYM
A: DUMPDISK CMD : DUMPDISK A86 : DUMPDISK BAK
A>
```

〔표 2〕 화면 1에 나타난 명령어들의 기능표

명령어	기능	해당하는 MS-DOS의 명령어
STAT	디스크나 화일, 시스템 등의 상태나 사용 상황을 표시하거나 변경함. (EX) A>STAT *.*	CHKDSK, DIR A>DIR *.*
PIP	디스크간의 화일 복사나 화일과 주변장치간의 데이터 통신을 위한 명령어 (EX) A>PIP B:=A: *.*(화일 보상)	COPY A>COPY A: *.* B:
ED	프로그램 개발용의 라인 에디터. (EX) A>ED TY.A86	EDLIN A>EDLIN TY.ASM
ASM86	8086 어셈블러 (EX) A>ASM86 TY	MASM A>MASM TY;
GENCMD	인텔 HEXA 화일을 명령어 화일로 전환시키는 로더. (EX) A>GENCMD TY	LINK A>LINK TY;
DDT86	프로그램 디버거. (EX) A>DDT86 TY.CMD	DEBUG A>DEBUG TY.EXE
SUBMIT	배치 명령어 (EX) A>SUBMIT BATCH(BATCH.SUB 화일에 명령어 리스트가 있어야 함).	배치화일
TOD	시각 및 날짜를 설정할 수 있는 유틸리티. (EX) A>TOD 06/14/1989 07:20:00	DATE, TIME

력을 한다면, 거의 모든 프로그램을 사용할 수 있다.

그러나 그렇게까지 되기에는 기본적으로 표 4에서 표시한 바와 같이 각기 대응하는 레지스터로 변환시켜 주고, 시스템 콜을 "CALL 0005H"에서 "INT 0E0H"로 변환시키면 가능하게 된다.

그러면 실제 예를 들어 살펴보자. 만일, 우리가 어떤 화일을 화면(콘솔)상에 표기하고자 한다면, 디스크 드라이브 콘트롤러에게 적당한 명령어를 주어 원하는 화일을 찾아서, 그 문자를 읽어들여 CRT 콘트롤러에 화면표시 방법을 일러준 후에야 그 초기화면을 디스플레이하는 것이 가능해진다.

그러나 일반적인 사용자가 이러한 복잡하고도 어려운 작업을 일일이 해준다면, 컴퓨터를 사용하고자 하는 사람은 아무도 없을 것이다. 따라서 CP/M과 같은 운영체계는 이러한 작업을 사용자가 편하게 할 수 있도록 배려하여, 사용자 인터페이스를 준비하였다. 그것이 바로 시스템 콜인 것이다. 이러한 전제를 바탕으로 표 5의 간단한 예제를 살펴보자.

표 5의 예는 문자 'A'를 콘솔로 출력하는 프로그램을 각 운영체계의 시스템콜에 맞추어서 표현한 것이다. 여기에서 볼 수 있듯이, CP/M-80과 CP/M-86은 표 4의 레지스터 변환 표에 따라 그대로 변환되었다는 것을 알 수 있다. 심지어는 MS-DOS에서조차 CP/M의 시스템콜을 배려한 흔적을 엿볼 수 있다.

따라서, CP/M-80에서 작성한 어셈블리 원시(Source) 화일을 적당한 변환 프로그램을 거쳐 고쳐주고 이것을 다시 ASM86으로 어셈블한다면, 그대로 CP/M-86의 시스템에서도 그 프로그램을 사용할 수 있게 된다. 이것을 도해한 것이 그림 3이다.

(3) 변환 프로그램 XLT-86

그림 3과 같은 변환을 위해 사용되는 프로그램이 XLT86이며, 이것은 8080의 어셈블리 프로그램을 8086용의 프로그램으로 자동변환시키려는 의도로 개발된 유틸리티이다.

그럼 그림 3의 흐름을 따라 가 보자. 먼저 변환시키고자 하는 XXX.ASM 화일을 XLT86의 입력으로 넣는다. 그렇게 하면, XLT86은 이 어셈블리 원시화일을 이용하여 그 프로그램의 전체적인 데이터의 흐름을 분석하여 적당한 8086의 구조로 바꾼다. XLT86의 생명은 이 "Global Data Flow Analysis(광역 자료 흐름의 분석)"이고 비슷한 종류의 변환프로그램에 비해 매우 효율이 좋은 구조를 발생시킨다. 이와 같이 변환된 화일은 XXX.A86이라는 화일명으로 저장된다.

이제 이 화일을 CP/M-86 기계에 삽입하여 ASM86.CMD와 GENCMD.CMD를 통해 실행화일을 만들면 8080상의 기계에서 실행되던 이 화일이 그대로 8086 기계에서도 수행되게 된다.

참고로 언급할 사항은 대부분의 CP/M 머신은 그 디스크 구조가 동일하며, 8비트에서 작성된 화일이라도, 16비트 시스템에서 그 화일을 그대로 액세스(Access) 할 수 있으나, 예외적으로 APPLE CP/M SofTCard를 사용하는 시스템은 이것이 불가능하다.

그것은 APPLE의 디스크시스템이 APPLE DOS에

〔표 3〕 CP/M-80과 CP/M-86의 차이점

	CP/M-80	CP/M-86
DIRS (.SYS 화일을 찾는다.)	없 음	추 가
SAVE	있 음	삭제 (대신 DDT86에 명령어 추가)
화일의 확장자	• COM • ASM • HEX • PRN • SYM	• CMD • A86 • H86 • LST • SYM
이름이 바뀐 외부 명령어	ASM. COM DDT. COM	ASM86. CMD DDT86. CMD
추가된 외부 명령어		HELP. CMD (각명령어에 대한 도움말을 줌) TOD. CMD (현재의 날짜/시간을 설정함)
I/O 기기의 명칭	CON : (콘솔) RDR : (리더) PUN : (펀치) LST : (프린터)	CON : (콘솔) AXI : (보조입력) AXO : (보조출력) LST : (프린터)

〔표 4〕 레지스터 대응표

CP/M-80 (8080/Z80 CPU)	CP/M-86 (8086계열 CPU)
A	AL
B	CH
C	CL
D	DH
E	DL
H	BH
L	BL
BC	CX
DE	DX
HL	BX

〔표 5〕 문자 'A'를 콘솔로 출력할 경우, 각 운영체계의 시스템콜

CP/M-80	CP/M-86	MS-DOS	의 미
LD C, 02H	MOV CL, 02H	MOV AH, 02H	← 함수 번호
LD A, 41H	MOV AL, 41H	MOV DL, 41H	← 전달 인수
CALL 0005H	INT 0E0H	INT 21H	← 시스템 콜

맞추어져 있어서 그 디스크구조가 상이하기 때문이다. 따라서, APPLE CP/M 사용자는 전술한 작업을 할 수 없으므로 널 모뎀이나, 특별한 하드웨어 장치를 통한 화일전송장치가 필요하게 된다.

다행히 시중에는 APPLE DOS와 IBM간의 통신용 카드가 있고, 또 최근에는 APPLE에서도 모뎀을 이용한 통신이 가능하므로 이를 통한 화일전송시스템을 구축하면 되겠다. 만일 하드웨어에 관심이 많은 독자라면 본지 1월호에 게재된 "애플과 IBM-PC와의 통신"에서 다룬 방법과 같이 별도의 통신 하드웨어를 제작하고 이에 맞는 프로그램을 작성한다면, 더 없는 큰 공부가 되겠다.

그리고 금성사의 8비트 마이티나 삼성의 8비트 SPC를 CP/M으로 운영하는 독자는 무리없이 변환이 가능하다.

CP/M-86의 내부구조

CP/M이 BIOS, BDOS, CCP, TPA로 이루어져 있다는 것은 앞에서도 언급한 바가 있다. 여기에서는 이들 내부 시스템구조에 관해 비교적 구체적으로 다루어 CP/M-86에 대한 전반적인 이해와 함께, 시스템 콜을 이용한 프로그램의 작성까지를 다루고자 한다.

(1) CP/M-86의 메모리 맵과 기동

IBM/PC상에 CP/M-86이 로드되었을 때의 메모리 구성과 시동시 초기 화면을 각각 **그림 4**와 **화면 2**에서 보였다.

CP/M-86이 기동되면 먼저 ROM BIOS에서는 절대 트랙 0, 섹터 1에서 부트스트랩 로더(Boot Strap Loader)를 메모리의 0000:7C00H로(640K 시스템에서) 업로드시킨다. 그런 후, 명령어 포인터 (CS:IP)를 0000:7C00H로 이동시켜 CP/M-86이 부트스트래핑을 하도록 유도한다. 이때 CP/M-86의 부트스트랩 로더는 디스크상의 0 트랙에서 4트랙까지를 주기억장치의 0051:0000H로 읽어들이고, 포인터를 바이오스의 콜드 스타트 어드레스(0051:2500H)로 옮겨 그 제어권을 바이오스에게 넘겨준다.

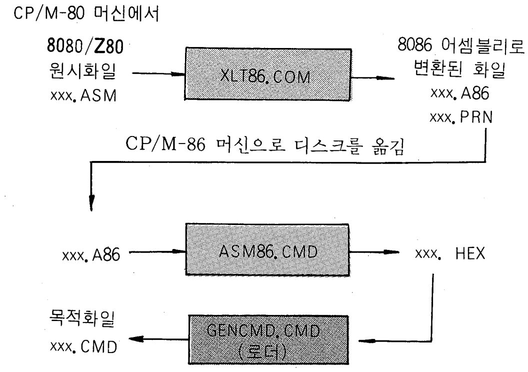
바이오스의 콜드스타트 함수는 기계상에 CP/M-86의 초기환경을 설정한 후, 그 제어권을 CCP에게 넘겨주어 사용자의 입력을 기다리게 된다. 이렇게 함으로써 CP/M-86의 부트가 완료된다. 참고로 **리스트 1**에 CP/M-86의 부트스트랩 로더를 실었다. 관심있는 독자는 참고하기 바란다(그림 5 참조).

(2) CP/M-86의 바이오스

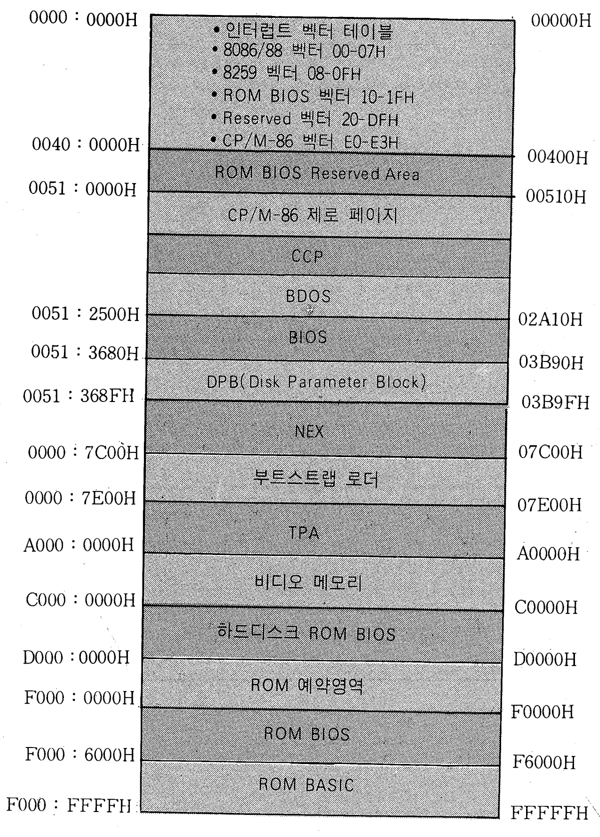
CP/M은 설계 당시부터 타기종으로의 이식성을 고려하여 작성된 운영체계이므로 CPU와 주변장치간의 입/출력을 행하는 바이오스의 소스를 공개하고 있다. 따라서, CP/M 머신의 설계자는 이 바이오스를 자신이 설계한 기계와 맞도록 고쳐야 한다. 이렇게 고쳐진 바이오스를 CBIOS(Customized BIOS)라 하며, 이것의 구성은 **그림 6**의 블록 다이어그램과 같다.

CP/M에서 디스크를 포함한 모든 입/출력은 BDOS에 의하고, 이 BDOS를 통해 행해진다. CP/M이 어떤 입/출력을 행할 때 BDOS는 바이오스의 선두에 설정되어 있는 점프 벡터에 의해 유도되므로, 그 점프 벡터의 번지를 콜하는 것으로 원하는 함수를 불러낼

〔그림 3〕 XLT86(8080→8086변환기)의 동작원리



〔그림 4〕 IBM/PC에서의 CP/M-86 메모리 맵



CP/M-86의 분석/특집

〔화면 2〕 초기화면

```
CP/M-86 Bootstrap Loader 1.0
Reading Track 0 1 2 3 4

CP/M-86 for the IBM Personal Computer.
Version 1.0
Copyright 1982, Digital Research Inc.


Hardware Supported :
                Diskette(s) : 2
                 Printer(s) : 1
              Serial Port(s) : 1
                Memory (Kb) : 128

A>

        3U=00302/10/82300:00:313
```

수 있게 된다.

또한 디스크 관련 함수가 불려졌을 때 BDOS는 디스크 파라미터 벡터를 바이오스 함수 9번 SELDSK에 의해 참조하고 디스크 관리에 필요한 각종 데이터가 들어 있는 번지를 구하여 그 디스크의 데이터를 얻음으로써 모든 디스크의 조작을 행한다. 따라서, 고쳐진 바이오스는 CP/M BDOS의 부함수(Sub Function)가 되므로 BDOS와 인터페이스하는 부분은 일정해야 하며, 이를 위해 제공되는 것이 바이오스 점프벡터 테이블이다.

CP/M에서 제공하는 BIOS함수는 모두 17개이며,

〔리스트 1〕 부트스트랩 로더

```
-10:7c00,7c94
0000:7C00 CLI
0000:7C01 CLD
0000:7C02 MOV   AX,CS     ┐
0000:7C04 MOV   DS,AX     │
0000:7C06 MOV   DX,0000   │ 세그먼트설정
0000:7C09 MOV   SS,DX     │
0000:7C0B MOV   SP,7C00   ┘
0000:7C0E STI
0000:7C0F MOV   SI,7C9A   ┐ 프린트 "CP/M-86 Bootstrap
0000:7C12 CALL  7C84      ┘         Loader 1.0"
0000:7C15 MOV   AX,0051   ┐
0000:7C18 MOV   ES,AX     │ 리세트 디스크
0000:7C1A MOV   AX,0000   │
0000:7C1D INT   13        ┘
0000:7C1F JB    7C7C
0000:7C21 MOV   AL,07     ┐
0000:7C23 MOV   BX,0000   │ 0051 : 0000H에서
0000:7C26 MOV   CX,0002   │ Track0, Sector 2~8리드
0000:7C29 CALL  7C61      ┘
0000:7C2C ADD   BX,0E00   ┐
0000:7C30 MOV   AL,04     │ 0051 : 0E00H에서
0000:7C32 MOV   CX,0105   │ Track1, Sector 5~8리드
0000:7C35 CALL  7C61      ┘
0000:7C38 ADD   BX,0800   ┐
0000:7C3C MOV   AL,08     │ 0051 : 1600H에서
0000:7C3E MOV   CX,0201   │ Track2, Sector 1~8리드
0000:7C41 CALL  7C61      ┘
0000:7C44 ADD   BX,1000   ┐
0000:7C48 MOV   AL,08     │ 0051 : 2600H에서
0000:7C4A MOV   CX,0301   │ Track3, Sector 1~8 리드
0000:7C4D CALL  7C61      ┘
0000:7C50 ADD   BX,1000   ┐
0000:7C54 MOV   AL,08     │ 0051 : 3600H에서
0000:7C56 MOV   CX,0401   │ Track4, Sector 1~8리드
0000:7C59 CALL  7C61      ┘
0000:7C5C CS:   JMPF  [7C95]
0000:7C61 PUSH  AX
0000:7C62 PUSH  BX        ┐
0000:7C63 MOV   AL,CH     │
0000:7C65 ADD   AL,30     │ 트랙넘버 프린트
0000:7C67 MOV   AH,0E     │
0000:7C69 MOV   BX,0000   │
0000:7C6C INT   10        ┘
0000:7C6E MOV   AX,0E20   ┐ 스페이스 프린트
0000:7C71 INT   10        ┘
0000:7C73 POP   BX
0000:7C74 POP   AX
0000:7C75 MOV   AH,02     ┐ 트랙 리드
0000:7C77 INT   13        ┘
0000:7C79 JB    7C7C      ← 에러
0000:7C7B RET
0000:7C7C MOV   SI,7CC9   ┐
0000:7C7F CALL  7C84      │ 에러루틴
0000:7C82 JMPS  7C15      ┘
0000:7C84 PUSH  BX        ┐
0000:7C85 LODSB           │
0000:7C86 OR    AL,AL     │
0000:7C88 JZ    7C93      │
0000:7C8A MOV   AH,0E     │ 에러메시지
0000:7C8C MOV   BX,0000   │
0000:7C8F INT   10        │
0000:7C91 JMPS  7C85      │
0000:7C93 POP   BX        ┘
0000:7C94 RET
```

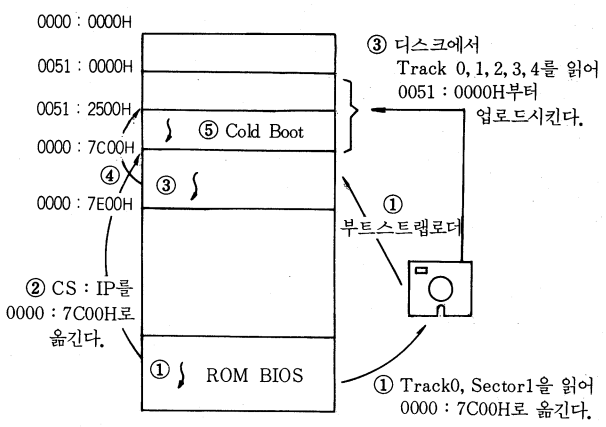
그 테이블은 표 6과 같다. 원래 이렇게 제공되는 BIOS 함수는 일반 사용자에게도 개방되어 있어서, 이를 BDOS를 거치지 않고도 직접 사용할 수 있게 되어 있다. 이것은 그림 7과 같은 흐름에 따라 사용자가 BIOS의 엔트리 포인트를 콜하는 것으로 가능하지만, CP/M-86에서는 이 BIOS 콜을 다른 형태로 제공하고 있다. 그것은 BDOS 시스템 콜에 바이오스에 대한 직접적인 서비스를 받을 수 있도록 함수가 준비되어 있다는 것이다.

그러면 왜 CP/M-86에서는 이러한 함수를 두고 있는 걸까? 그것은 8086/88 마이크로프로세서의 세그먼트와 오프셋이라는 번지지정방식(Addressing)으로 인해 바이오스를 직접 부를 경우, 이 루틴은 그림 7과 같이 종국에는 RET를 만나게 된다. 그러나 이것은 64K 내에서만 사용되는 것이어서 시스템은 스택의 세그먼트를 POP하지 않게 되어 BIOS를 부른 프로그램으로 되돌아 올 수 없게 된다.

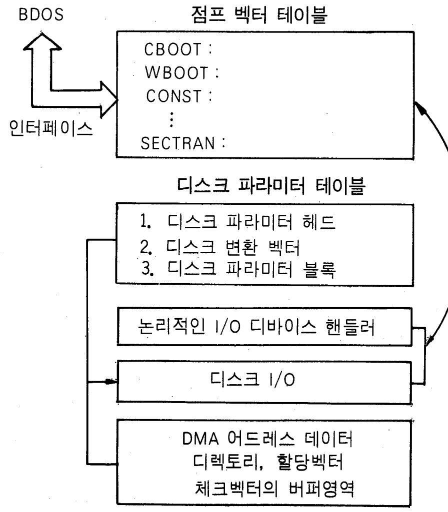
만일 우리가 BIOS로부터 직접 입력을 받아 그것을 콘솔로 출력하는 프로그램을 리스트 2와 같이 작성하였다고 하자. 이 프로그램의 첫번째 라인에서는 BIOS의 CONIN 루틴을 FAR CALL로 불렀으므로 스택상에 이것이 리턴한 다음에 되돌아 올 번지를 저장하고 CONIN 루틴으로 점프하게 된다. 물론 이 루틴은 정상적으로 수행되어 화면상에서 한문자를 기다리게 될 것이다.

이때, 우리가 키보드를 통해서 입력을 준다면 다음의 CONOUT 루틴이 수행되어야 하나 실제로는 그림 7과 같이 "RET"("RETF"가 아니라)를 만나게되므로 엉뚱한 곳으로 코드세그먼트가 설정되어 다시 되돌아

〔그림 5〕 CP/M-86의 부트스트랩 과정



〔그림 6〕 CBIOS의 구성



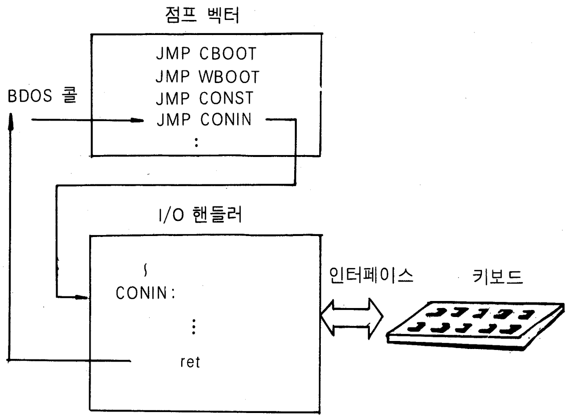
〔표 6〕 바이오스 점프벡터의 엔트리포인트

번호	번 지	벡터명	기 능	전달 인수	리턴 값
0	0051 : 0000	CBOOT	콜드부트(Cold Boot)	—	CL=0
1	0051 : 0003	WBOOT	웜부트(Warm Boot)	—	CL=DRIVE
2	0051 : 0006	CONST	콘솔 상태 체크	—	AL=FF/00
3	0051 : 0009	CONIN	콘솔로부터 입력	—	AL=CHAR
4	0051 : 000C	CONOUT	콘솔로 출력	CL=출력값	—
5	0051 : 000F	LIST	프린터로 출력	CL=출력값	—
6	0051 : 0012	AUXOUT	보조출력장치로 출력	CL=출력값	—
7	0051 : 0015	AUXIN	보조입력장치에서 입력	—	AL=CHAR
8	0051 : 0018	HOME	디스크 헤드의 홈 SEEK	—	—
9	0051 : 001B	SELDSK	디스크의 선택	CL=DRIVE	BX=DPB
10	0051 : 001E	SETTRK	트랙의 설정	CL=TRACK	—
11	0051 : 0021	SETSEC	섹터의 설정	CL=SECTOR	—
12	0051 : 0024	SETDMA	DMA 번지설정	CX=DMA	—
13	0051 : 0027	READ	지정된 섹터의 읽기	—	AL=00/01
14	0051 : 002A	WRITE	지정된 섹터로 쓰기	—	—
15	0051 : 002D	LISTST	프린터 상태 체크	—	AL=FF/00
16	0051 : 0030	SECTRAN	섹터 변환	CX=SECTOR DX=TRABLE	BX=물리적 SECTOR NO

을 수가 없게 되는 것이다. 그것은 "RET"가 스택상에서 단지 2바이트만을 POP하기 때문이다.

이제 이것을 BDOS콜의 50(32H)을 이용하여 BIOS를 불러내는 방법을 살펴보자. 리스트 3은 BDOS 콜을 이용한 BIOS의 이용 예를 보인 것으로 이는 리스트 2의 의도와 같은 것이다. 이 프로그램에서는 바이오스에 전달할 인수들을 바이오스 디스크립터라는 것을 통해서 하게 되는데, 그 내용은 함수번호 CX, DX로 구성된 5바이트의 것을 의미하며, 리스트 3에서는 PARA1, PARA2가 그것을 지칭하게 된다. 이 프로그램에서는 그림 7과 같이 정상적인 바이오스 콜을 BDOS가 행해주고 본 프로그램으로는 "IRET"로 돌아오므로 세그먼트 관계가 흐트러지지 않게 되는 것이다.

〔그림 7〕 BIOS의 흐름도



〔리스트 2〕

```
START  CALLF  0051:2509   ; CALL CONIN

       MOV    CL,AL       ;
       CALLF  0051:250C   ; CALL CONOUT
```

〔리스트 3〕

```
START:  MOV   CL,3H              ; 함수 번호
        MOV   DX,OFFSET PARA1    ; 전달 인수 엔트리
        INT   0E0H               ; BDOS콜을 불러냄

        MOV   DX,OFFSET PARA2+1  ;
        MOV   [DX],AL            ; 키보드로부터 얻은 값을 저장
        DEC   DX                 ; BIOS DESCRIPTOR 설정
        MOV   CL,4H              ;
        INT   0E0H

        ...

PARA1   DB      03H              ; BIOS 함수 번호
        DW    0000H              ; CX
        DW    0000H              ; DX

PARA2   DB      04H              ; BIOS 함수 번호
        DW    0000H              ;
        DW    0000H              ;
```

그러면 오히려 BDOS함수를 호출하는 것보다 더 복잡한 방법을 써가면서 BIOS 콜을 호출하는 것은 무슨 이유 때문일까? 그것은 BDOS의 다른 함수에서 제공하지 않는 미세한 디스크운영을 할 수 있기 때문이다. 예를 들면, 디스크 에디터와 같이 디스크상의 특별한 부분을 고치고자 할 때는 이 바이오스콜이 효율적이기 때문이다.

리스트 4는 직접 바이오스콜을 사용한 예제로써, 디스크상의 섹터를 십육진 및 아스키 문자로 덤프하는 프로그램이다. 이 프로그램은 물론 예제의 성질도 있겠지만, 다음에서 설명할 디스크구조에 대한 하나의 도구로써 제작한 것이다. 또한 이것은 조금만 고치면 디스크 에디터로도 사용할 수 있으므로 관심있는 독자의 도전을 바라는 바이다.

이 프로그램에 대한 어셈블과정과 그 사용 예는 화면 3과 같다.

(3) BDOS 시스템 콜

시스템 콜이라고 하는 것은 CP/M이 갖고 있는 각종 주변장치와의 입출력기능이나, 디스크 화일의 조작 등의 기능을 사용자 프로그램에서 사용하도록 허용하는 것으로 이는 운영체계가 사용자에게 제공하는 시스템서비스이다. 이 시스템 콜은 "펑션 콜(Function Call)" 또는 "수퍼바이저 콜(Supervisor Call)"이라고도 불리우며, CP/M에서 가장 중요한 부분이다. CP/M이 최근까지 마이크로 컴퓨터의 표준 운영체계로 자리잡을 수 있던 것은 바로 이러한 서비스가 사용자에게 쉬운 형태로 제공되었기 때문이다. MS-DOS가 CP/M 계열의 운영체계라고 하는 것도, 이러한 시스템 서비스를 CP/M과 거의 같게 제공해 주기 때문이다.

CP/M-86에서 시스템콜을 부르는 방법은, 먼저 CL 레지스터에 원하는 기능을 가진 함수번호를 넣고, 표 7에 나타낸 인수값을 해당 레지스터에 넣은 다음, "INT 0E0H" 명령어를 수행시켜 시스템콜을 부르는 것으로 족하다.

표준 CP/M의 경우, 시스템 콜 함수의 수는 40(28H)번까지 있으나, CP/M-86에서는 50번부터 10개의 함

수가 추가되었다. 이것은 8086계열의 CPU가 1MB의 메모리공간을 갖고 있으므로 이것을 관리하기 위해 추가된 것이다. 따라서, 64KB를 넘는 프로그램을 작성하고자 할 경우, 이 함수들을 적절하게 이용하면 효율적인 프로그램이 가능하게 된다.

이제, CP/M-80이나 MS-DOS의 시스템 콜을 사용해본 경험이 없는 독자를 위해 가벼운 예제를 들어서 시스템콜의 사용방법을 설명해 보고자 한다.

리스트 5는 "type" 명령어와 꼭 같은 역할을 하는 프로그램을 시스템 콜을 사용하여 나타낸 것으로 단지 80 바이트 정도의 명령어를 사용하여 그 기능을 충분히 나타낼 수 있음을 보여주고 있다.

"TY.A86"이란 이름으로 작성한 화일을 ASM86을 통해 어셈블하여 얻어진 리스팅 화일이다. 먼저, 100H 번지에서 106H번지까지는 FCB에 등록된 화일을 여는 (Open)부분으로 함수 15번을 사용하였다. 이 함수는 정상적으로 화일을 열었을 경우에는 AL 레지스터에 디렉토리 코드를 저장시키고, 그렇지 않을 경우에는 AL레지스터에 0FFH 코드를 돌려줌으로써 화일을 열 수 없음을 표시해 준다. 따라서, 이 때는 에러 메시지를 출력하고 프로그램을 빠져 나간다.

다음 112H에서 117H까지는 이렇게 연 화일을 80H 바이트 만큼 읽어들여 디폴트 DMA 버퍼인 CS:0080H로 업로드한다. 그리고나서 이 DMA에 저장된 데이터를 함수 2번을 사용하여 콘솔로 출력한다. 이 부분을 "END OF FILE"이 될 때까지 반복하는 것이 이 프로그램의 전부이다.

따라서 "type"명령어를 구성하기 위해서는 시스템 콜의 함수 2번, 9번, 15번, 20번 등의 4개를 사용하면 간단히 구성할 수 있게 된다.

(4) CP/M-86의 화일시스템

BDOS는 디스크시스템을 관리하기 위해서 DPB(Disk Parameter Block)이라는 테이블을 참조하여 디스크조작을 행하고 있다. 여기에는 리스트 6에서 나타낸 디스크 관련 인자가 기록되어 있는데, 이 각각에 대해 자세히 설명을 한다는것은 다소 지루하고 전문적인 내용이어서 다음기회로 미루고 간략하게 개념만을 짚고 넘어가고자 한다.

먼저 SPT란 한트랙을 몇개의 트랙으로 나누어서

〔화면 3〕 어셈블과정과 디스크 덤프프로그램 사용 예

```
A>
A>asm86 dumpdisk

CP/M 8086 ASSEMBLER VER 1.1
END OF PASS 1
END OF PASS 2
END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR:  1%

A>gencmd dumpdisk

BYTES READ    037B
RECORDS WRITTEN 0A


A>dumpdisk

DISK DUMP UTILITY PROGRAM
   -PROGRAMMED BY KIM,JS
    JUN 13,1989

     DISK : 00        TRACK : 00        SECTOR : 00
------------------------------------------------------------------
0000  +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
------------------------------------------------------------------
0000  FA FC 8C C8 8E D8 BA 00 00 8E D2 BC 00 7C FB BE   z¦.H.X:...R<.¦{>
0010  9A 7C E8 6F 00 B8 51 00 8E C0 B8 00 00 CD 13 72   .¦ho.8Q..@8..M.r
0020  5B B0 07 BB 00 00 B9 02 00 E8 35 00 81 C3 00 0E   [0.;..9..h5..C..
0030  B0 04 B9 05 01 E8 29 00 81 C3 00 08 B0 08 B9 01   0.9..h)..C..0.9.
0040  02 E8 1D 00 81 C3 00 10 B0 08 B9 01 03 E8 11 00   .h...C..0.9..h..
0050  81 C3 00 10 B0 08 B9 01 04 E8 05 00 2E FF 2E 95   .C..0.9..h....
0060  7C 50 53 8A C5 04 30 B4 0E BB 00 00 CD 10 B8 20   ¦PS.E.04.;..M.8
0070  0E CD 10 5B 58 B4 02 CD 13 72 01 C3 BE C9 7C E8   .M.[X4.M.r.C>I¦h
D)ISK   * : EXIT   LEFT/RIGHT : SECTOR   UP/DN : TRACK   HOME/ENDC
     DISK : 00        TRACK : 01        SECTOR : 01
------------------------------------------------------------------
0000  +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D-+E +F
------------------------------------------------------------------
0000  00 44 44 54 38 36 20 20 20 43 4D 44 00 00 00 6D   .DDT86   CMD...m
0010  19 1A 1B 1C 1D 1E 1F 00 00 00 00 00 00 00 00 00   ................
0020  00 41 53 53 49 47 4E 20 20 43 4D 44 00 00 00 10   .ASSIGN  CMD....
0030  20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ...............
0040  00 4E 45 57 44 49 53 4B 20 43 4D 44 00 00 00 32   .NEWDISK CMD...2
0050  21 22 23 24 00 00 00 00 00 00 00 00 00 00 00 00   !"#$............
0060  00 46 55 4E 43 54 49 4F 4E 43 4D 44 00 00 00 0E   .FUNCTIONCMD....
0070  25 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   %...............

D)ISK   * : EXIT   LEFT/RIGHT : SECTOR   UP/DN : TRACK   HOME/END
```

입출력을 하는가라는 것으로 대개의 경우 (디스크 시스템에 따라 다르지만) 트랙당 32섹터가 보통이다. 따라서, IBM-PC의 5인치 플로피 디스크에서는 한트랙

〔표 7〕 CP/M-86의 시스템콜 일람표

함수 번호		함수의 기능	전달인수	리턴 값
십진	HEX			
0	00	시스템 리세트	DL=종료 코드	없음
1	01	키보드로부터의 입력	없음	AL=입력 문자
2	02	화면으로의 출력	DL=출력문자	없음
3	03	AUX로부터의 입력	없음	AL=입력문자
4	04	AUX로의 출력	DL=출력문자	없음
5	05	프린터로의 출력	DL=출력문자	없음
6	06	직접적인 콘솔 입출력	입력 DL = 0FFH	AL=입력문자
			입력 DL = 0FFH	AL=키보드상태
			DL=출력문자	없음
7	07	IO 바이트 취득	없음	AL=IO 바이트
8	08	IO 바이트 세트(Set)	DL=IO 바이트	없음
9	09	문자열을 콘솔로 출력	DS:DX=문자열 번지	없음
10	0A	버퍼로 키보드 입력	DS:DX=버퍼번지	〔버퍼〕=문자열 〕
11	0B	키보드 상태체크	없음	AL=키보드상태
12	0C	CP/M의 버전 획득	없음	BX=버전 NO
13	0D	디스크의 리세트	없음	없음
14	0E	드라이브의 선택	DL=드라이브 NO	없음
15	0F	화일을 연다(Open)	DS:DX=FCB 번지	AL=에러코드
16	10	화일을 닫는다(Close)	DS:DX=FCB 번지	AL=에러코드
17	11	최초의 화일 탐색	DS:DX=FCB 번지	AL=00/FF
18	12	다음 화일의 탐색	없음	AL=00/FF
19	13	화일의 삭제	DS:DX=FCB 번지	AL=00/FF
20	14	순차화일 읽기	DS:DX=FCB 번지	AL=00/FF
21	15	순차화일 쓰기	DS:DX=FCB 번지	AL=00/FF
22	16	화일의 생성	DS:DX=FCB 번지	AL=00/FF
23	17	화일이름의 변경	DS:DX=FCB 번지	AL=00/FF
24	18	로그인 벡터의 취득	없음	BX=로그인 벡터
25	19	로그인 디스크의 취득	없음	AL=디스크 NO
26	1A	DMA번지설정	DS:DX=DMA 번지	없음
27	1B	할당 번지취득	없음	ES:BX=할당번지
28	1C	쓰기방지의 설정	없음	없음
29	1D	R/O벡터 취득	없음	BX=벡터, AL=Dir
30	1E	화일속성의 설정	DS:DX=FCB 번지	없음
31	1F	DPB번지취득	없음	ES:BX=DPB 번지
32	20	유저코드 설정 및 취득	DL=사용자코드/0FFH	AL=사용자코드
33	21	랜덤하게 읽기	DS:DX=FCB 번지	정상 : AL=00
34	22	랜덤하게 쓰기	DS:DX=FCB 번지	에러 : AL=에러코드
35	23	화일크기의 계산	DS:DX=FCB 번지	FCB:r0-r2=크기
36	24	랜덤레코드의 세트	DS:DX=FCB 번지	FCB:r0-r2=레코드
37	25	디스크의 리세트	DX=드라이브벡터	AL=00
38	26	—		
39	27	—		
40	28	랜덤하게 라이트(ZeroFile)	DS:DX=FCB 번지	AL= 코드
50	32	직접 바이오스 호출	DX=BIOS 디스크립터	

〈뒤 계속〉

을 8섹터로 나누고 있기 때문에 실제의 물리적 섹터와는 차이점을 갖고 있다. 그래서 CP/M은 이 물리적인 섹터와 논리적인 CP/M의 섹터를 적당히 맵핑시켜서 실제의 디스크 입출력이 어떠하든 관계없이 똑같은 알고리즘으로 처리하고 있다.

BSF는 데이터의 저장 단위로써 128바이트의 2의 BSF승 즉, 리스트 6에서는 128*2̂4=2048바이트가 디스크에 저장되는 최소단위가 된다. 따라서, CP/M의 경우는 디스크손실이 크게 나타난다. 극단적인 예로써 1바이트의 화일들로도 디스크를 차게 할 수 있는 것이다. 이러한 개념으로 저장하는 시스템은 블록이라는 용어를 사용한다. 이때, 2048바이트가 하나의 블록이 되고 1바이트든지 2048바이트든지 디스크에 저장될 때는 1블록이 된다.

BLM은 한 블록의 경계가 되는 섹터의 수를 말하며, 여기서는 0에서 15섹터가 한 블록이 되므로, BLM의 값은 15가 된다.

DSM은 데이터블록의 최대수를 의미하며, 여기서는 0에서 156블록이 있음을 나타낸다. (따라서 대략 157*2-048=320K 바이트 정도가 디스크용량임을 알 수 있다.)

DRM은 최대 화일 수-1로, CP/M에서는 최대화일을 64개로 하고 있다. AL0, AL1은 디렉토리 엔트리가 사용하는 데이터블록을 표시한 것으로 다음과 같은 비트패턴에 따라 결정된다. 리스트 6의 AL0, AL1이 각각 10H와 00H이므로 그 비트 패턴은 그림 8과 같다. 그림 10에서 1비트만 세트되어 있으므로 1개의 데이터 블록 즉 2048바이트가 디렉토리영역으로 사용되고 있음을 보여준다.

이러한 기본 개념을 갖고 CP/M의 화일처리 시스템에 대해 살펴보자. 우선 CP/M의 화일을 취급하기 위해서는

함수 번호		함수의 기능	전달 인수	리턴 값
십진	HEX			
51	33	DMA 세그먼트 설정	DX= Base address	없음
52	34	DMA 세그먼트 취득	없음	ES:BX=DMA 번지
53	35	가용 메모리	DX= offset MCB	AL=코드
54	36	최대 메모리취득	〃	〃
55	37	메모리영역 획득	〃	〃
56	38	절대메모리의 획득	〃	〃
57	39	메모리영역 릴리즈	〃	〃
58	3A	전메모리 해방	〃	〃
59	3B	프로그램의 로드	〃	〃

(참고) MCB (Memory Control Block) 란 CP/M-86에서 메모리공간을 1MB까지 이용하기 위해 사용되는 것으로 다음과 같은 형식을 취하고 있다.

MCB :	M-Base	M-Length	M-Ext
	15 … 0	15 … 0	7 … 0

여기서 M-Base는 메모리영역의 베이스 어드레스를 표시하고, M- Length는 영역의 길이를 표시하며, M-Ext는 리턴코드를 나타낸다.

〔그림 8〕 AL0와 AL1의 비트패턴

AL0								AL1							
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

〔그림 9〕 실제 FCB

```
┌ 드라이브                          화일의 크기(총레코드수=128×6D바이트)
(00) 44 44 54 38 36 20 20 20 43 4D 44 00 00 00 (6D)  .DDT86   CMD...m
     └──────── 화일명 ───────┘ └ 확장자 ┘

19 1A 1B 1C 1D 1E 1F 00 00 00 00 00 00 00 00 00   ...............
└ 1K단위의 디스크위치 ┘

   랜덤 액세스시 사용
(00) 00 00 00                                       ......
└ 현재레코드 포인터
```

① 화일의 쓰기/읽기는 1레코드(128바이트) 단위로 이루어지며,
② FCB와 DMA라는 화일 입출력 관련 영역에 대한 이해가 필요하며,
③ 화일을 쓰고/읽기 위해서는 그 화일을 열어야 하며,
④ 화일을 쓰고 난 다음에는 반드시 그 화일을 닫아 주어야 한다는 점

등에 유의하여야 한다.

FCB는 화일의 이름과 그 화일의 디스크상에서의 위치를 기록해둔 디스크상의 화일정보영역을 의미하며 대개 33바이트 또는 36바이트가 필요하다. 그림 9에 보인 FCBS는 디스크상의 FCB를 시스템이 메모리로 복사한 것이다. 디스크상의 FCB는 트랙 1, 섹터 0부터 있는데, 그림 9를 통해서 그를 확인할 수 있다.

CP/M상에서 화일을 취급하려면 이 화일에 대한 정보를 메모리로 로드시켜서 그것을 이용하여야 한다. 이때, FCB를 미리 설정하지 않는다면, CP/M은 디폴트 FCB영역인 커런트(Current) 세그먼트의 005CH로 그 화일정보를 로드시킨다. 이때, 로드되는 FCB의 상태는 그림 9와 같으며 이러한 기능을 이용해서 작성한 것이 "TY.A86"의 프로그램이다.

즉,

```
A>TY  TY.A86
```

이라는 명령어를 CCP에 주었을 때, CCP는 이를 해석하여 TY.CMD를 TPA의 상위번지를 로드시키고, 따라오는 어큐먼트 TY.A86은 디폴트 FCB인 CS:005CH로 TY.A86에 대한 디스크 FCB를 복사하게 된다. 따라서, 사용자는 이를 염두에 두지 않고 단지 어큐먼트가 CS:005CH로 올라온다고 생각해도 무방한 것이다. 만일 두개의 어큐먼트를 연속해서 사용한다면, 다음의 디폴트 FCB인 CS:006CH로 그것을 복사하게 된다.

그리고, 화일을 읽거나 쓰고자 할 경우는 그 내용을 어떤 일정한 버퍼에 저장해 두고 일정한 크기가 되었을 때 한꺼번에 입출력을 하는 것이 효율적이다. 이를 위해 준비된 것이 DMA버퍼라는 것이다. 따라서 화일 입출력에는 반드시 이 DMA에 관한 어드레스를 기억할 필요가 있는데, 그 디폴트 DMA 버퍼가 CS:0080H이다. 이 DMA는 시스템 콜을 통해서 변경시킬 수 있다. 리스트 5의 "TY.A86"에서는 디폴트 DMA를 사용하고 있으므로 CS:0080H를 DMA의 선두 어드레스로 생각하여 읽어들인 화일을 이곳에서 찾아 화면으로 표시한다. 이때, 사용되는 DMA 버퍼의 크기는 0080H(128) 바이트이며, 이 크기는 CP/M의 1 논리섹터의 크기이다.

화일을 쓰고/읽기 위해서는 그 화일의 FCB에 실제의 디스크상의 위치를 설정해야 하는데 이러한 과정의 처음단계를 화일의 열기(Open)라 한다. 이를 위해 제공되는 시스템함수가 15번이고 다음과 같은 표현으로 가능하다.

```
MOV   CL,15          ; 화일 열기 함수 번호
```

```
	MOV	DX,OFFSET FCB	; FCB의 선두 번지
	INT	BDOS		; 시스템 콜 호출
```

물론, "TY.A86"에서도 이와 같은 열기과정을 거쳤음은 앞에서도 언급했다. 반대로 화일을 닫는 것은 함수번호 16번을 이용해서 열 때와 똑 같이 해주면 된다. 만일 열거나 닫을 화일이 존재하지 않으면, 시스템은 AL레지스터에 0FFH를 보내 준다. 따라서 이 리턴값을 갖고 판단하면, 쉽게 화일의 유무를 알 수 있다.

이제 화일을 읽기 위해서는 시스템 콜 20번을 이용한다.

```
	MOV	CL,20		; 함수 번호
	MOV	DX,OFFSET FCB	; FCB 선두 번지
	INT	BDOS		;
```

이와 같이 하면, 화일의 읽혀진 내용이 디폴트 DMA인 CS：0080H로 로드되고 이를 시스템 콜 2번을 이용하여 프린트한 것이 앞에서 든 예제 프로그램이다.

```
	MOV	BX,OFFSET DMA	; BX = DMA 선두 번지
L1:	PUSH	BX		; BX 저장
	MOV	AL,[BX]		; DMA에서 한 문자를 얻음
	MOV	DL,AL		;
	MOV	CL,02		; 함수 2 번 호출
	INT	BDOS		; 콘솔로 프린트
	POP	BX		; BX RESTORE
	INC	BX		; BX 증가
	CMP	BL,0		; BL<>0 면 루프
	JNZ	L1		;
```

MS-DOS와 CP/M-86

1980년대 중반이후 사용자층을 갖게된 MS-DOS는 앞에서도 언급한 바가 있지만, CP/M 계열의 운영체계로서 탄생했고, 그 설계개념이 CP/M을 모체로 하고 있는 것은 주목할 만한 사실이다. 물론, MS-DOS는 계층적 디렉토리구조와 미래를 향한 멀티태스킹에 대한 배려, 그리고 화일시스템에 이르기까지 UNIX의 구조를 함께 포함하고 있는 것 또한 사실이다. 그러나 대부분의 명령어와 시스템 콜의 처리방법 등은 확실히 CP/M의 것과 유사하다. 따라서, CP/M 사용자가 쉽게 MS-DOS에 적응할 수 있는 것도 이와 같은 유사성 때문이다.

이러한 유사성은 시스템 콜의 호출에서도 발견된다. MS-DOS에서 제공하는 시스템 콜은 세가지 방법을 통해 수행시킬 수 있다. 그 첫번째 방법은 대표적으로 사용하는 INT 21H를 이용하는 방법이며, 두번째 방법은 CALL 0005H를 사용하는 것이며, 마지막 하나는 CALL PSP+0050H를 사용하는 것이다. 여기서 두번째의 시스템 콜에 대한 사용법은 기존의 많은 CP/M 사용자를 끌어들이기 위한 전략에서 나온 것으로 그 내용이 거의 CP/M의 그것과 똑 같다.

또한 이 방법은 CP/M-80이 00H에서 24H의 함수를 지원하는 것과 마찬가지로 MS-DOS가 더 많은 함수를 준비하였음에도 불구하고, 이 함수이상은 지원하지 않고 있다. 이것을 통해서도 MS-DOS가 기존의 CP/M 사용자를 얼마나 의식하고 만들어졌는지를 쉽게 알 수 있다. 따라서 MS-DOS는 함수를 부를 때 그 함수번호가 단지 CL 레지스터에서 AH 레지스터로 바뀌었다는 점과 시스템호출을 INT 21H를 통해서 한다는 점이므로, 서로의 화일을 적당한 변환을 통해 재구성한다면, 하나의 수행화일을 똑같이 수행시킬 수 있게 된다.

실제로 MS-DOS와 CP/M-86은 같은 트랙구조를 갖고 있으므로, 각 디스크시스템을 잘 파악하여 서로의 화일을 한 기계에서 교환할 수 있도록 프로그램을 작성할 수 있다.

그렇지만, MS-DOS가 그 화일체계에 있어서 CP/M과 UNIX를 모방하여 사용하고 있을지라도 그것을 저장하는 시스템은 CP/M의 그것과는 사뭇 다르다. 그것은 전술한 바와 같이 CP/M에서는 블록이라는 개념으로 디스크 입출력을 행하고 있지만, MS-DOS에서는 클러스터라는 개념을 도입해 좀 더 복잡한 처리과정을 통해서 화일을 액세스하고 있다. 이러한 방법의 도입으로 MS-DOS에서는 1바이트단위의 화일관리가 가능하게 되었다. 따라서 CP/M의 디스크저장 시스템보다 더욱 효율적으로 설계되어 있다. 그러나 이러한 상이함으로 인해, 같은 머신상에서도 서로 화일전송이 이루어지지 않게 되었다. 즉 이것을 가능하게 하기 위해서는 한 시스템에서 다른 시스템의 운영체계가 하는 것과 똑같은 작업을 해주어야 하므로 그 변환 작업이 쉬울리 없겠다.

CP/M상에서 수행되는 응용 프로그램

CP/M은 약15년간을 개인용 컴퓨터의 가장 각광받는 운영체계로서 군림해온 관계로, 엄청나게 많은 응용프로그램을 갖고 있다. 최근의 MS-DOS에서 수행되는 수 많은 프로그램이 이를 바탕으로 작성된 것은 물론이다.

CP/M을 개발하거나 수정하기 위해 사용되는 어셈

블러로는 ASM이외에도, MAC, RMAC, MACRO80 등이 있으며, 언어패키지로는 COBOL, FORTRAN, BASIC, C, PASCAL, PL / I, PL / M, FORTH, LISP, ALGOL, APL, PROLOG 등을 총 망라하고 있으며, 그밖에 SuperCalc나 VisiCalc와 같은 전자종이 및 WordStar, WordMaster 등의 스크린 에디터에 이르기까지 매우 많이 개발되어 있다. 또한 CP / M 머신은 타 기계의 개발용으로도 많이 사용되어 왔으므로 그러한 기기 개방용 소프트웨어도 다양하다.

결어

지금까지 CP / M-86의 특징 및 그 내부구조 그리고 CP / M의 전부라 할 수 있는 BDOS 시스템 콜에 관해서 설명하였다. 원래 필자의 의도는 CP / M-86의 시스템레벨에서 접근하여 일반 사용자도 어셈블리어에 대한 지식만 있으면 누구나 쉽게 CP / M 프로그램을 작성하고 초보자에게는 이러한 운영체계가 현재 사용되는 대부분의 운영체계에 있어서 모범이 되고 있음을 보여 주고자 했다. 그러나 이에 대한 설명이 부족하지 않았나 생각한다.

어쨌든 이 글을 읽은 독자들이 CP / M-86에 대해 관심을 갖기를 바라며 계속되는 운영체계의 발전과정에서 CP / M의 자리매김이 어떻게 이루어지는지를 살펴 봐 주기 바란다.

기회가 주어진다면 CP / M-86용 디스크에디터나 CP / M-86과 MS-DOS간의 화일전송 등에 관한 프로그램을 작성하여 CP / M-86의 구체적인 응용예로써 또는 상이한 운영체계간의 차이점과 그 극복방법에 대해 다루고자 한다. 이글을 끝까지 읽어준 독자 여러분들에게 감사드리며 글을 갈음하고자 한다. PC

〔리스트 4〕 디스크 덤프프로그램

```
                        ;
                        ;       DISK DUMP PROGRAM
                        ;
                        ;               PROGRAMMED BY KIM,JONGSEOK
                        ;
00E0                    BDOS    EQU     0E0H    ; BDOS 인터럽트
0009                    SELDSK  EQU     09H     ; BIOS의 디스크기능 선택
000A                    SETTRK  EQU     0AH     ; BIOS의 트랙넘버 기능 세트
000B                    SETSEC  EQU     0BH     ; BIOS의 섹터넘버 기능 세트
000C                    SETDMA  EQU     0CH     ; BIOS의 DMA 버퍼기능 세트
000D                    READ    EQU     0DH     ; BIOS의 섹터기능 리드
                        ;
                        ;
                                CSEG            ; 세그먼트 선언
                                ORG     100H
                        ;
                        ;
0100 B109               START:  MOV     CL,09H  ; 헤딩메시지(BDOS) 프린트
0102 BACA02                     MOV     DX,OFFSET HEADING
0105 CDE0                       INT     BDOS
                        ;
                        LOOP:
0107 E87B01     0285            CALL    STATUS  ; 현재상태
010A E8AD00     01BA            CALL    DUMP    ; 디스크덤프
                        ;
010D B109                       MOV     CL,09H  ; 프롬프트 메시지 프린터
010F BA1303                     MOV     DX,OFFSET PROMPT
0112 CDE0                       INT     BDOS
                        ;
0114 B101               RDKEY:  MOV     CL,01H  ; BDOS를 사용한 콘솔로부터 한문자 가져옴
0116 CDE0                       INT     BDOS    ;
                        ;
0118 3C2A                       CMP     AL,'*'  ;
011A 742B       0147            JZ      TERMINATE
                        ;
011C 3C44                       CMP     AL,'D'  ;
011E 742A       014A            JZ      SELDISK
                        ;
0120 3C44                       CMP     AL,'D'  ;
0122 7426       014A            JZ      SELDISK
```

```
                   ;
0124 3C1B                  CMP    AL,1BH   ; ESC 시퀀스
0126 75EC      0114        JNZ    RDKEY
0128 B101                  MOV    CL,01H   ; 한문자 입력
012A CDE0                  INT    BDOS
                   ;
012C 3C41                  CMP    AL,41H   ;UP키
012E 7432      0162        JZ     DECTRK
                   ;
0130 3C42                  CMP    AL,42H   ; DN키
0132 7422      0156        JZ     INCTRK
                   ;
0134 3C44                  CMP    AL,44H   ; LEFT키
0136 7442      017A        JZ     DECSEC
                   ;
0138 3C43                  CMP    AL,43H   ; RIGHT키
013A 7432      016E        JZ     INCSEC
                   ;
013C 3C48                  CMP    AL,48H   ;HOME키
013E 7446      0186        JZ     HOME
                   ;
0140 3C4E                  CMP    AL,4EH   ; END키
0142 7451      0195        JZ     ENDTRK
                   ;
0144 E9CDFF    0114        JMP    RDKEY
                   TERMINATE:
0147 E9D900    0223        JMP    EXIT
                   ;
                   ;
014A BBB101        SELDISK:MOV    BX,OFFSET DISK
014D 8A07                  MOV    AL,[BX]
014F 3401                  XOR    AL,01H
0151 8807                  MOV    [BX],AL
0153 E9B1FF    0107        JMP    LOOP
                   ;
0156 BBA701        INCTRK: MOV    BX,OFFSET TRACK
0159 8A07                  MOV    AL,[BX]
015B FEC0                  INC    AL
015D 8807                  MOV    [BX],AL
015F E9A5FF    0107        JMP    LOOP
                   ;
0162 BBA701        DECTRK: MOV    BX,OFFSET TRACK
0165 8A07                  MOV    AL,[BX]
0167 FEC8                  DEC    AL
0169 8807                  MOV    [BX],AL
016B E999FF    0107        JMP    LOOP
                   ;
016E BBAC01        INCSEC: MOV    BX,OFFSET SECTOR
0171 8A07                  MOV    AL,[BX]
0173 FEC0                  INC    AL
0175 8807                  MOV    [BX],AL
0177 E98DFF    0107        JMP    LOOP
                   ;
017A BBAC01        DECSEC: MOV    BX,OFFSET SECTOR
017D 8A07                  MOV    AL,[BX]
017F FEC8                  DEC    AL
0181 8807                  MOV    [BX],AL
0183 E981FF    0107        JMP    LOOP
                   ;
0186 32C0          HOME:   XOR    AL,AL          ; AL=0
0188 BBA701                MOV    BX,OFFSET TRACK
018B 8807                  MOV    [BX],AL
018D BBAC01                MOV    BX,OFFSET SECTOR
0190 8807                  MOV    [BX],AL
0192 E972FF    0107        JMP    LOOP
                   ;
                   ENDTRK:
0195 B04F                  MOV    AL,79
0197 BBA701                MOV    BX,OFFSET TRACK
019A 8807                  MOV    [BX],AL
019C B01F                  MOV    AL,31
019E BBAC01                MOV    BX,OFFSET SECTOR
01A1 8807                  MOV    [BX],AL
01A3 E961FF    0107        JMP    LOOP
                   ;
                   ;
01A6 0A            DTRACK  DB     SETTRK
01A7 00            TRACK   DB     00
01A8 00                    DB     00
01A9 00                    DB     00
01AA 00                    DB     00
```

```
01DA BB8000                MOV   BX,0080H
01DD 53            DLOOP:  PUSH  BX
01DE B000                  MOV   AL,0
01E0 E87300    0256        CALL  TOHEX
01E3 8AC3                  MOV   AL,BL
01E5 247F                  AND   AL,7FH
01E7 E86C00    0256        CALL  TOHEX
01EA E84D00    023A        CALL  SPC
01ED E84A00    023A        CALL  SPC
01F0 B91000                MOV   CX,0010H
01F3 8A07          DLP:    MOV   AL,[BX]
01F5 E85E00    0256        CALL  TOHEX
01F8 E83F00    023A        CALL  SPC
01FB 43                    INC   BX
01FC E2F5      01F3        LOOP  DLP
                   ;
01FE E83900    023A        CALL  SPC
0201 E83600    023A        CALL  SPC
0204 5B                    POP   BX
0205 B91000                MOV   CX,0010H
0208 8A07          ALP:    MOV   AL,[BX]
020A 8AD0                  MOV   DL,AL
020C E81800    0227        CALL  ASCII
020F 43                    INC   BX
0210 E2F6      0208        LOOP  ALP
                   ;
0212 E83000    0245        CALL  CRLF
0215 81FB0001              CMP   BX,0100H
0219 72C2      01DD        JB    DLOOP
021B C3                    RET
                   ;
                   ;
021C BA5703        ERROR:  MOV   DX,OFFSET ERRMSG
021F B109                  MOV   CL,09H
0221 CDE0                  INT   BDOS           ; 에러 메시지 프린트 출력
                   ;
0223 B100          EXIT:   MOV   CL,00H         ;
0225 CDE0                  INT   BDOS           ; EXIT기능.
                   ;
                   ;
0227 53            ASCII:  PUSH  BX
```

```
                   ;
01AB 0B            DSECTOR DB    SETSEC
01AC 00            SECTOR  DB    00
01AD 00                    DB    00
01AE 00                    DB    00
01AF 00                    DB    00
                   ;
01B0 09            DDISK   DB    SELDSK
01B1 00            DISK    DB    00
01B2 00                    DB    00
01B3 00                    DB    00
01B4 00                    DB    00
                   ;
01B5 0D            DREAD   DB    READ
01B6 00                    DB    00
01B7 00                    DB    00
01B8 00                    DB    00
01B9 00                    DB    00
                   ;
                   ;
                   DUMP:
                   ;
01BA B132                  MOV   CL,50
01BC BAB001                MOV   DX,OFFSET DDISK
01BF CDE0                  INT   BDOS           ; 디스크 선택
                   ;
01C1 B132                  MOV   CL,50
01C3 BAA601                MOV   DX,OFFSET DTRACK
01C6 CDE0                  INT   BDOS           ; 트랙 선택
                   ;
01C8 BAAB01                MOV   DX,OFFSET DSECTOR
01CB B132                  MOV   CL,50
01CD CDE0                  INT   BDOS           ; 섹터 선택
                   ;
01CF BAB501                MOV   DX,OFFSET DREAD
01D2 B132                  MOV   CL,50
01D4 CDE0                  INT   BDOS           ; 섹터 리드
                   ;
01D6 3C00                  CMP   AL,00H         ; 에러
01D8 7542      021C        JNZ   ERROR
                   ;
```

```
0228 51                     PUSH   CX
0229 247F                   AND    AL,7FH
022B 3C20                   CMP    AL,' '
022D B22E                   MOV    DL,'.'
022F 7202       0233        JB     DOT              ; 콘솔문자→'.'
0231 8AD0                   MOV    DL,AL            ;
0233 B102             DOT:  MOV    CL,02H           ;
0235 CDE0                   INT    BDOS             ;
0237 59                     POP    CX
0238 5B                     POP    BX
0239 C3                     RET
                      ;
                      ;
                      ;
023A 53               SPC:  PUSH   BX
023B 51                     PUSH   CX
023C B102                   MOV    CL,02H
023E B220                   MOV    DL,' '
0240 CDE0                   INT    BDOS
0242 59                     POP    CX
0243 5B                     POP    BX
0244 C3                     RET
                      ;
                      ;
0245 53               CRLF: PUSH   BX
0246 51                     PUSH   CX
0247 B102                   MOV    CL,02H
0249 B20D                   MOV    DL,0DH
024B CDE0                   INT    BDOS
                      ;
024D B102                   MOV    CL,02H
024F B20A                   MOV    DL,0AH
0251 CDE0                   INT    BDOS
0253 59                     POP    CX
0254 5B                     POP    BX
0255 C3                     RET
                      ;
                      ;
0256 53               TOHEX: PUSH  BX
0257 51                     PUSH   CX
0258 50                     PUSH   AX
0259 24F0                   AND    AL,0F0H
025B D0E8                   SHR    AL,1
025D D0E8                   SHR    AL,1
025F D0E8                   SHR    AL,1
0261 D0E8                   SHR    AL,1
0263 3C09                   CMP    AL,9
0265 7602       0269        JBE    DECI1
0267 0407                   ADD    AL,07
0269 0430           DECI1:  ADD    AL,48
026B B102                   MOV    CL,02H
026D 8AD0                   MOV    DL,AL
026F CDE0                   INT    BDOS

0271 58                     POP    AX
0272 240F                   AND    AL,0FH
0274 3C09                   CMP    AL,9
0276 7602       027A        JBE    DECI2
0278 0407                   ADD    AL,07
027A 0430           DECI2:  ADD    AL,48
027C B102                   MOV    CL,02H
027E 8AD0                   MOV    DL,AL
0280 CDE0                   INT    BDOS

0282 59                     POP    CX
0283 5B                     POP    BX
0284 C3                     RET
                    ;
                    ;
0285 BA7C03         STATUS: MOV    DX,OFFSET DISKID
0288 B109                   MOV    CL,09H
028A CDE0                   INT    BDOS
                    ;
028C BBB101                 MOV    BX,OFFSET DISK
028F 8A07                   MOV    AL,[BX]
0291 E8C2FF     0256        CALL   TOHEX
                    ;
0294 E8A3FF     023A        CALL   SPC
0297 E8A0FF     023A        CALL   SPC
                    ;
029A BA8C03                 MOV    DX,OFFSET TRACKID
029D B109                   MOV    CL,09H
```

```
                              ;
 0313 0D0A44294953     PROMPT         DB      0DH,0AH,'D)ISK   * : EXIT   LEF
T/RIGHT : SECTOR  UP/DN : TRACK   HO
      4B2020202A20
      3A2045584954
      2020204C4546
      542F52494748
      54203A205345
      43544F522020
      2055502F444E
      203A20545241
      434B20202048
      4F4D452F454E
      4424
 0357 4449534B2049     ERRMSG         DB      'DISK IS UNABLE TO READ THIS SE
CTOR',0DH,0AH,'$'
      5320554E4142
      4C4520544F20
      524541442054
      484953205345
      43544F520D0A
      24
                              ;
 037C 0D0A20202020     DISKID         DB      0DH,0AH,'      DISK : $'
      20204449534B
      203A2024
 038C 202020202020     TRACKID        DB      '       TRACK : $'
      545241434B20
      3A2024
 039B 202020202020     SECTORID       DB      '       SECTOR : $'
      534543544F52
      203A2024
 03AB 2D2D2D2D2D2D     HEADER         DB      '-------------------------------
--------------------------------------
      2D2D2D2D2D2D
      2D2D2D2D2D2D
      2D2D2D2D2D2D
      2D2D2D2D2D2D
      2D2D2D2D2D2D
      2D2D2D2D2D2D
      2D2D2D2D2D2D
```

```
 029F CDE0                        INT     BDOS
                         ;
 02A1 BBA701                      MOV     BX,OFFSET TRACK
 02A4 8A07                        MOV     AL,[BX]
 02A6 E8ADFF       0256           CALL    TOHEX
                         ;
 02A9 E88EFF       023A           CALL    SPC
 02AC E88BFF       023A           CALL    SPC
                         ;
 02AF BA9B03                      MOV     DX,OFFSET SECTORID
 02B2 B109                        MOV     CL,09H
 02B4 CDE0                        INT     BDOS
                         ;
 02B6 BBAC01                      MOV     BX,OFFSET SECTOR
 02B9 8A07                        MOV     AL,[BX]
 02BB E898FF       0256           CALL    TOHEX
                         ;
 02BE E884FF       0245           CALL    CRLF
                         ;
 02C1 BAAB03                      MOV     DX,OFFSET HEADER
 02C4 B109                        MOV     CL,09H
 02C6 CDE0                        INT     BDOS
 02C8 C3                          RET
 02C9 C3                          RET
 02CA 0D0A4449534B       HEADING          DB      0DH,0AH,'DISK DUMP UTILITY PROG
RAM'
      2044554D5020
      5554494C4954
      592050524F47
      52414D
 02E5 0D0A2020202D                        DB      0DH,0AH,'   -PROGRAMMED BY KIM,
JS'
      50524F475241
      4D4D45442042
      59204B494D2C
      4A53
 02FF 0D0A20202020                        DB      0DH,0AH,'    JUN 13,1989',0DH,0
AH,'$'
      4A554E203133
      2C313938390D
      0A24
```

CP/M-86의 분석/특집

```
                   2D2D2D2D2D2D
                   2D2D2D2D2D2D
                   2D2D2D2D2D2D
                   2D2D2D2D2D2D
                   0D0A
              03F5 303030302020              DB      '0000  +0 +1 +2 +3 +4 +5 +6 +7
+8 +9 +A +B +C +D +E +F',0DH,0AH
                   2B30202B3120
                   2B32202B3320
                   2B34202B3520
                   2B36202B3720
                   2B38202B3920
                   2B41202B4220
                   2B43202B4420
                   2B45202B460D
                   0A
              042C 2D2D2D2D2D2D              DB      '-----------------------------
-----------------------------------------
       2D2D2D2D2D2D      ---'
       2D2D2D2D2D2D
       2D2D2D2D2D2D
       2D2D2D2D2D2D
       2D2D2D2D2D2D
       2D2D2D2D2D2D
       2D2D2D2D2D2D
       2D2D2D2D2D2D
       2D2D2D2D2D2D
       2D2D2D2D2D2D
       2D2D2D2D2D2D
       0D0A24

END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR:  1%
```

〔리스트 5〕 TY.A86 소스프로그램

```
                        ;       FILE TYPE PROGRAM, READS AN INPUT FILE AND PRIN
T OUT TO CONSOLE
                        ;
                        ;       PROGRAMMED BY J.S.KIM
                        ;                    1989.6.11
                        ;
                        ;       TO CMD FILE GENERATION, ISSUE THE NEXT COMMAND
                        ;
                        ;       A>ASM86 TY
                        ;       A>GENCMD TY
                        ;
                        ;
 00E0                   BDOS    EQU     0E0H    ; BDOS 인터럽트넘버
                        ;
 005C                   FCB     EQU     05CH    ; 화일 콘트롤 블록
 0080                   BUFFER  EQU     080H    ; 버퍼프린트 엔트리
                        ;
                        ;
 000D                   CR      EQU     0DH     ; 캐리지 리턴
 000A                   LF      EQU     0AH     ; 라인 피드

                ;
                        CSEG            ; 코드 세그먼트 할당
                        ORG     0100H   ; 스타트번지 설정
                ;
0100 B10F       MAIN:   MOV     CL,0FH  ; 콜넘버=15 기능
0102 BA5C00             MOV     DX,FCB  ; DX=FCB
0105 CDE0               INT     BDOS    ; 시스템콜
                ;
0107 3CFF               CMP     AL,0FFH ; 에러 오픈?
0109 742B    0136       JZ      OPENERR ; Yes
                ;
010B B000               MOV     AL,00H  ;
010D BB7C00             MOV     BX,FCB+32
0110 8807               MOV     [BX],AL
                ;
                NEXTRECORD:
0112 B114               MOV     CL,14H  ; 화일을 순차적으로 읽기.
0114 BA5C00             MOV     DX,FCB  ;
0117 CDE0               INT     BDOS    ;
                ;
```

```
0119 3C00                CMP   AL,00H  ;
011B 7515   0132         JNZ   EXIT    ; 읽기 성공
                 ;
011D BB8000              MOV   BX,BUFFER
0120 B102         PRINT: MOV   CL,02H  ;콘솔로 프린트 출력
0122 8A17                MOV   DL,[BX] ; 한문자 전송
0124 53                  PUSH  BX      ;
0125 CDE0                INT   BDOS    ;
0127 5B                  POP   BX      ;
                 ;
0128 43                  INC   BX      ;
0129 8AC3                MOV   AL,BL   ;
012B 3C00                CMP   AL,00H  ;
012D 75F1   0120         JNZ   PRINT   ;

012F E9E0FF 0112         JMP   NEXTRECORD
                 ;
0132 B100         EXIT:  MOV   CL,00H  ; 시스템 리세트
0134 CDE0                INT   BDOS    ;
                 ;
                 OPENERR:
0136 B109                MOV   CL,09H  ; 에러메시지 프린트 출력
0138 BA4001              MOV   DX,OFFSET MSGERR
013B CDE0                INT   BDOS
013D E9F2FF      0132    JMP   EXIT
                 ;
0140 0D0A46494C45 MSGERR DB    CR,LF,'FILE NOT FOUND',CR,LF,'$'
     204E4F542046
     4F554E440D0A
     24
                 ;
                         END

END OF ASSEMBLY.  NUMBER OF ERRORS:  0.  USE FACTOR:  0%
```

〔리스트 6〕 데이터 파라미터블록의 예

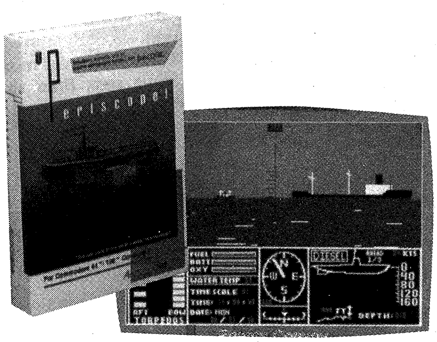
```
DPB     EQU     $

SPT     DW      0020H   ; SECTOR PER TRACK ( 1트랙당 섹터 수 )
BSF     DB        04H   ; BLOCK SHIFT FACTOR ( BLS = 128 * 2^BSF )
BLM     DB        0FH   ; BLOCK MARK ( 1 BLOCK = 0 - 15 SECTOR )
EXM     DB        01H   ; EXTENT MARK
DSM     DW      009DH   ; DATA SIZE MAX(DISK BLOCK SIZE의 최대수 - 1)
DRM     DW      003FH   ; DIRECTORY ENTRY MAX( 최대 가능 화일 수 = 64 개 )
AL0     DB        10H   ; DATA BLOCK의 비트패턴
AL1     DB        00H   ;
CKS     DW      0010H   ; CHEXK VECTOR SIZE
OFF     DW      0001H   ; TRACK OFFSET
```

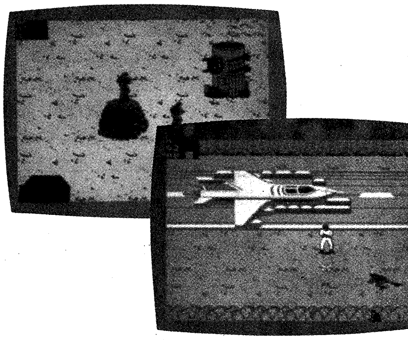
cf) 이것은 5 인치 양면 배밀도 디스크의 경우를 나타낸 것임.



4×4 OFF-ROAD PACING



UPERISCOPE



AIRBORNE RANGER

10TH FRAME PROBOWLING	1
221B BAKER STREET	2
2400 AD	1
3D HELICOPTER	1
4TH & INCHES	1
ACE OF ACE	1
AUTO DUEL	2
AIRBORNE RANGER	1
BAD STREET BRAWER	1
BADUK	1
BALANCE OF POWER	3
BARD'S TALE Ⅰ	2
BARD'S TALE Ⅱ	2
BATTLE CHESS	2
BATTLE ZONE	1
BATTLEHAWKS 1942	2
BIONIC COMMANDO	1
BLACK CAULDRON	2
BLACK JACK ACAEMY	1
BLOCK BUSTER	1
BOP,N WRESTLE	1
BORROWED TIME	1
BUBBLE GHOST	1
CALIFONIA GAME	1
CARD SHARKS	1
CHAMPIONSHIP BASEBALL	1
CHAMPIONSHIP BOXING	1
CHAMPIONSHIP GOLF	1
CHUCK YEAGER'S FLIGHT TRAINNER	1
COMMANDO	1
CONTRA	1
CRAZY CARS	1
CRIME & PUNISHMENT	1
DAM BUSTER	1
DARK CASTLE	2
DEATH SWORD	1
DEFENDER OF THE CROWN	1
DESTROYER	1
DOUBLE DRAGON	2
DOWNHILL CHALLENGE	2
DRAGON WORLD	2
EARL WEAVER BASE BALL	2
ECHELON	2
ELITE	1
EMPIRE	1
F-15 STRIKE EAGLS	1
F-19 STEALTH FIGHTER	3
FALCON F-16	1
FIRE POWER	2
FLIGHT SIMULATOR	1
FLIGHT SIMULATOR V3.0	2
FORD SIMULATOR	1
FRIENDLY WARE GAME	1
FUTURE MAGIC	2
GAMESTAR BASEBALL	1
GATO	1
GAUNTLET	1
GHOSTS'N GOBLINS	1
GOLD RUSH	5
GRAND PRIX CIRCUIT	2

GUERRILA	2
HACKER Ⅱ	1
HARD BALL	1
HARRIER	2
HEROES OF LANCE	3
HULK	1
HUNT FOR RED OCTOBER	1
IKARI WARRIOR	1
IMAGE TENNIS	1
IMPOSSIBLE MISSION Ⅲ	1
INDOOR SPORTS	2
INFILTLATOR	1
INTO THE EAGLE'S NEST	1
JACK NIKLAUS GOLF	2
JEOPARDY	1
JEOPARDY 88	1
JET	1
KARNOV	2
KING'S QUEST Ⅰ	1
KING'S QUEST Ⅱ	2
KING'S QUEST Ⅲ	3
KING'S QUEST Ⅳ	9
KNIGHT ORC	2
LAST NINJA	2
LEISURE SUIT LARRY Ⅱ	6
LOCK ON	2
LODE RUNNER	1
MAD SPY VS SPY Ⅱ	1
MAN HUNTER	5
MANIAC MANSION	1
MARBLE MADNESS	2
MASTER NINJA	1
MIGHT & MAGIC	2
MINI PUTT	1
MIXED UP MOTHER GOOSE	2
MOEBIUS	2
NFL CHALLENGE	2
NINJA	1
OFF SHORE WARRIOR	1
ONE ON ONE	1
PATTON VS ROMMEL	1
PETE ROSE PENANT FEVER	2
PHANTASIE	1
PHANTASIE Ⅲ	1
PHYCHO	2
PINBALL WIZARD	1
PIRATES	2
PLATOON	2
POLICE QUEST Ⅱ	3
POLICE QUEST	6
POOL OF RADIANCE	3
PT-109	1
QUSTRON Ⅱ	2
RAD WARRIOR	1
RAMPAGE	1
RINGS OF ZILFIN	2
ROADWAR 2000	1
ROCK FORD	2
ROCKET RANGER	1
ROCKET RANGER	2
RACKEN	1

RUSH'N ATTACK	1
SCRABBLE	1
SDI	1
SEARCH AND DISTROY	1
SEVEN CITY OF GOLD	1
SHADOW GATE	2
SHANGHAI	1
SHOGUN	1
SIDE ARMS	1
SKATE OR DIE	2
SKY FOX Ⅱ	1
SKY RUNNER	1
SPACE QUEST Ⅰ	2
SPACE QUEST Ⅱ	3
SPACE QUEST Ⅲ	6
SPORTING NEWS BASEBALL	1
STAR COMMAND	3
STAR FLEET Ⅰ	1
STAR FLIGHT	2
STAR GLIDER	1
STAR TREK	1
STREET FIGHTER	2
SUB BATTLE	1
SUMMER GAME Ⅱ	1
SUPERBYKE CHALLENGE	1
SUPERSTAR ICEHOCKEY	1
TEST DRIVE	1
TETRIS	1
THE FELLOWSHIP OF THE RING	1
THE GANES SUMMER GAME EDITION	4
THE LAND OF THE LOUNGE LIZARDS	2
THEXDER	2
THREE STOOGES REEL	2
TKO	1
TOMAHAWK	1
TOP GUN	1
TREASURE ISLAND	2
TROJAN	1
TWILIGHT RANSOM	2
ULTIMA Ⅰ	1
ULTIMA Ⅱ	3
ULTIMA Ⅲ	2
ULTIMA Ⅳ	2
UNDER FIRE	2
VIDEO VEGAS	1
WASTELAND	5
WHERE IN THE EUROPE IS CARMEN	1
WHERE IN THE USA IS CARMENSAN	1
WHERE IN THE WOLRD IS CARMEN	1
WHO FRAMED ROGER RABBIT	2
WAR IN THE MIDDLE EARTH	3
WINTER GAMES	1
WIZARD WARS	2
WIZARDRY Ⅰ	2
WIZARDRY Ⅲ	1
WIZARDRY Ⅳ	1
WIZARDRY Ⅴ	3
WORLD GAME	1
WORLD GREATEST BASEBALL	1
WORLD KARATE CHAMPIONSHIP	1
WORLD TOUR GOLF	1

시스템 프로그램의 응용

김용득 / 아주대 전자과 교수

시스템의 구성체계

사용자가 컴퓨터로 주어진 업무를 처리하는 과정은 우선 컴퓨터가 이해할 수 있는 언어로 처리과정을 작성하고 이를 기억장소에 저장한 후 컴퓨터에 의해서 이를 읽어 처리한 다음 출력 매개체를 통하여 사용자에게 전달하는 여러 단계를 거치게 된다.

이러한 처리과정은 그림 1에서와 같이 소프트웨어부, 하드웨어부, 제어부로 구성된다. 소프트웨어부는 주어진 프로그램 언어를 사용하여 처리될 업무의 순서를 작성하는 것으로 그림 1에 명기된 것처럼 컴퓨터 시스템 분야의 5% 정도에 해당된다. 여기서 응용 프로그램이란 주어진 업무를 컴퓨터와 대화가 가능한 언어로 작성한 것을 말한다.

일반적으로 컴퓨터 언어에는 BASIC, FORTRAN, PASCAL, C언어 등과 같은 고급언어와 "1"과 "0"의 조합으로 구성(이를 기계어라 부름)되어 직접 컴퓨터 내부의 하드웨어를 작동시킬 수 있는 하급언어, 그리고 이 고급언어와 하급언어의 중간단계로서 명령어 형태를 단위 요약어로 표시하는 어셈블리어 등이 있다.

여기서 고급언어는 인간의 사용언어와 아주 유사하나 컴퓨터를 직접 구동시킬 수는 없다. 따라서 기계가 이해할 수 있는 1이나 0으로 구성된 기계어 프로그램으로 바꾸어 주어야 하는데 이 과정을 수행하는 것을 번역기(Translator) 라 부른다.

번역기에는 번역하는 방법에 따라 컴파일러(Compiler)와 인터프리터(Interpreter) 로 나뉘는데 이는 사용하는 언어에 따라서 앞에 그 언어명을 붙여 부른다. 즉 사용되는 언어가 FORTRAN 인 경우에 그 번역기는 FORTRAN 컴파일러라고 부르고, C언어이면 C컴파일러라 부른다. 특히 BASIC 언어인 경우에는 BASIC 인터프리터라고 명명되고 있다.

●컴파일러와 인터프리터의 비교

컴파일러나 인터프리터는 고급 컴퓨터 언어를 1과 0의 기계어로 변환해주는 번역기로 하나의 명령어가 하나의 기계어로 1:1 대응되는 것이 아니고 기계어로 구성된 단위 프로그램 그룹을 지정시킨다는 점에서 기본 개념은 같다. 그러나 이들은 번역된 것을 어떤 방법으로 처리하느냐에 따라 구별된다.

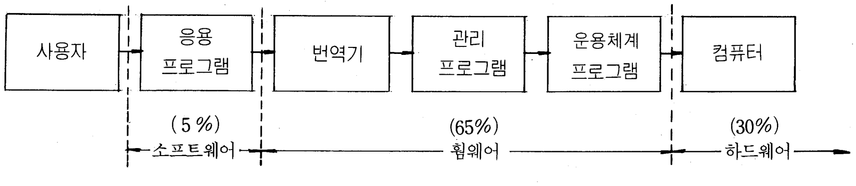
즉 컴파일러는 사용자가 작성하여 기억장소에 저장시킨 고급언어로 된 모든 프로그램(이를 소스(Source)프로그램이라 부름)을 동시에 번역하여 또 다른 번지의 기억장소에 저장한 후 (이들 프로그램을 오브젝트(Object)프로그램이라 부름) 이를 처리한다. 반면에 인터프리터는 소스 프로그램을 번역하여 기억장소에 저장하지 않고 매번 한 명령어씩 읽어 처리한다.

따라서 컴파일러는 한번에 번역하여 처리하므로 처리속도는 빠르다는 장점을 갖고 있으나 별도의 번역된 프로그램을 저장시켜야 되기 때문에 많은 양의 기억장소를 필요로 한다. 이러한 이유로 기억장소에 구애받지 않는 대형 컴퓨터에 적합한 방식이라 볼 수 있다.

인터프리터의 경우에는 한 명령어씩 읽어 번역한 후 이를 처리하는 방식이므로 번역된 프로그램을 저장시킬 별도의 기억장소가 필요없으나 시간이 많이 걸린다는 단점이 있다. 즉 반복 프로그램인 경우 매번 인터프리터에 의해 번역되어 처리되기 때문에 많은 시간이 소요된다. 따라서 마이크로프로세서와 같은 기억소자가 제한받는 소형 컴퓨터에 적합하다고 본다.

이와같이 고급언어는 컴파일러나 인터프리터에 의해 단위 기계어 프로그램으로 변환되지만 어셈블리어와 같은 기호화된 프로그램(Symbolic Program) 은 1:1로 대응되는 기계어 프로그램으로 대치되는데, 이때 사용되는 번

〔그림 1〕시스템 구성체계



역기를 어셈블러라 한다.

●크로스 어셈블러와 셀프어셈블러의 비교

어셈블러는 컴파일러방식과 같이 한번에 모두 번역하여 기억시킨 후에 이를 처리한다. 어셈블러 번역기의 구성에 관해서는 뒤에서 세부적으로 설명하겠지만 어떤 마이크로프로세서에 적용하여 번역시키느냐에 따라 크로스어셈블러(Cross-Assembler)와 셀프어셈블러(Self-Assembler)로 분류된다.

즉 크로스어셈블러는 CPU가 다른 기종에서 번역시키는 방법을 사용하며, 셀프어셈블러는 같은 종류의 CPU를 사용하여 번역시키는 것을 말한다. 예를 들어 IBM-PC상에서 자일로그사(Z80) 또는 모토롤라 계열(M68000 등)의 프로세서로 기호화된 어셈블리어를 번역시키고자 하는 경우에 사용되는 방식을 크로스어셈블러라 하고, 인텔의 8086/286 마이크로프로세서의 어셈블리어를 번역하고자 하는 경우에 사용하는 방식을 셀프어셈블러라 부른다.

참고로 잘 작동되고 있는 기존 시스템에 의하여 개발될 특정 프로세서시스템의 동작 프로그램만을 시험 검사하기 위한 방식을 시뮬레이션(Simulation) 이라 부르며, 개발된 하드웨어와 관련하여 하드웨어 및 동작프로그램을 동시에 검사 측정하기 위한 방식을 에뮬레이션(Emulation)이라 하고, 마이크로프로세서 시스템에 의하여 타 시스템을 에뮬레이션 처리할 수 있는 장치를 인서킷 에뮬레이터(In-Circuit Emulater)라 한다.

이들 장비는 새로운 마이크로프로세서를 사용한 시스템 설계에서 필수적으로 요구되는 것으로서 총괄적으로 MDS(Microprocessor Development System)라 부른다.

사용자 프로그램(Source Module)이 번역기를 통해 기계어 프로그램(Object Module)이 생성되면, 링크 프로그램에 의하여 이들 모듈화된 프로그램과 내부 프로그램(Library Program)이 결합되어 하나의 수행 프로그램이 형성되고 그 다음 이 수행 프로그램은 로더 프로그램에 의하여 실 기억장소에 저장하게 된다.

이들 로더, 링커 등과 같이 사용자 프로그램을 제어하기 위한 프로그램을 관리 프로그램(Debugger)이라 하며 이들은 그림 2와 같이 구성되어 있다.

시스템 운용체계

특정번지에 저장된 수행 프로그램은 입·출력 주변기기와 프로세서가 함께 작동되어 처리되어야 하는데, 이때 하드웨어를 구동시키기 위한 프로그램 그룹을 운용체계(OS : Operating System) 라 부른다. 이 운용체계는 하드웨어 제어프로그램 즉 모니터 프로그램으로서 프로세서의 프로그램 모듈의 제어관리, 입출력 장치의 제어관리, 기억장치의 제어관리가 포함된다.

따라서 컴퓨터를 개발한다면 기계부를 구성하는 하드웨어부(4회까지는 하드웨어부 설계에 관해서 설명하였음)와 운용체계가 포함된 훰웨어부(firmware) 가 연구 개발되어야 한다. 개발비용은 하드웨어부 개발에 10%, 훰웨어부 개발에는 90%로 대부분의 개발비용을 훰 웨어부가 차지하고 있다.

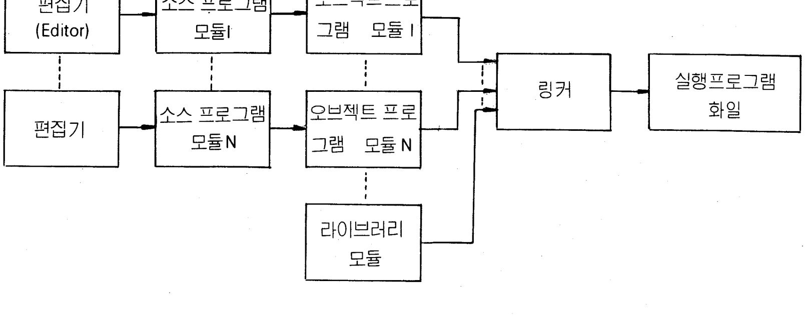
하드웨어부는 이미 마이크로프로세서(8, 16, 32, 64비트 등)가 VLSI화되어 생산되고 있고, 훰웨어 또한 표준화되어 있기 때문에 현재 마이크로프로세서 시스템을 개발한다고 할 때에는 단지 이들을 효율적으로 결합시키는 작업이라 할 수 있다.

훰웨어 중 표준화된 운용체계로는 인텔이나 자일로그사의 8비트 프로세서에 적용되는 CP/M(Control Program for Microprocessor)이 1970년대 후반에 개발되었으며 이를 수정보완하여 16비트 및 32비트 프로세서에 적용이 가능한 CP/M-86, Concurrent-CP/M 등이 발표되었고, 모토롤라 계열에서는 OS-9(6809계열), FLEX 등이 표준화되어 있다.

또한 중·대형 범용 컴퓨터에서 주로 사용되어 오던 Bell 연구소의 UNIX 및 마이크로소프트사의 ZENIX가 16비트 프로세서에서도 사용할 수 있도록 변형되었으며, 이들을 복합 수정하여 1984년 IBM-PC에 채택된 MS-DOS는 소형 컴퓨터 운용체계에서 과히 대표적이라 할 수 있다. 특히 소형 컴퓨터 즉 16비트 마이컴에 현재 이들의 사용실태는 표 1과 같이 MS-DOS의 사용이 우세하나, 최근에는 OS-2의 발전도 눈에 띈다.

그리고 그림 1과 같은 시스템 구성체계를 이해하므로써 컴퓨터 시스템설계와 구성을 할 수 있다. 따라서 이번에는 이들 컴퓨터가 어떻게 작동되어 주어진 프로그램을 처리하는가를 설명

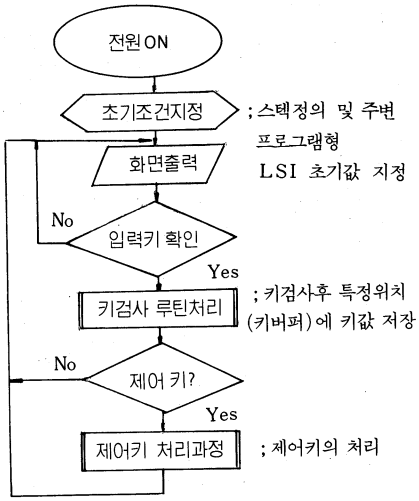
〔그림 2〕 관리프로그램의 운영



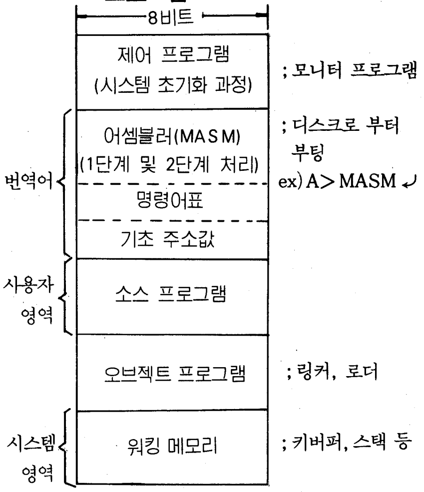
〔표 1〕 표준 OS의 비교

표준 OS의 이름	점유율
CP/M-86	5%
(Concurrent CP/M	5%
UNIX	10%
MS-DOS	60%
Custom	10%
기타	10%

〔그림 3〕 시스템 동작과정



〔그림 4〕 기억소자 영역에서 본 시스템 프로그램



하겠다(그림 3의 시스템 동작과정 참조).

전원이 인가되면 마이크로프로세서와 주변 입·출력기기를 제어할 프로그램형 LSI소자에 초기값을 지정해 주어야 하며(이는 다음호의 프로그램형 접속소자편에서 설명함), 이를 ROM에 내장된 모니터 프로그램에서 지정하게 된다. IBM-PC에서는 이를 ROM BIOS라 하며 ROM으로부터 초기값을 읽어 프로그램형 LSI 소자에 기억(Cold Start)시키면 이들 LSI는 하드디스크 또는 플로피디스크에 저장된 시스템 운용 프로그램을 내부 주기억소자에 이동(Warm Start)시키고, 화면상에는 사용자가 프로그램을 입력시켜 처리할 수 있는 준비단계가 완료되었음을 가리킨다.

그리고 키가 입력되면 입력된 키값을 키버퍼(Key Buffer)라는 특정 위치에 저장시키고 화면에 이를 표시한다. 이러한 동작은 리턴키(Carrage Return) 를 인지할 때까지 반복한다. 리턴키가 입력되면 입력된 값(One Line)들은 해석되고 그에 해당되는 기능을 수행하게 된다.

예를 들어 입력된 값이 편집기의 일종인 WS(Word Star)라면, 즉 "A>WS"라면 디스크에 기억된 WS 프로그램을 불러 내부 주기억소자에 저장시키고 지금까지 시스템 프로그램이 관할하던 하드웨어 제어 운영권을 WS라는 프로그램이 담당하게 된다.

WS에서는 여러분이 현재 사용하는 것과 같은 문자의 편집, 수정, 인쇄 등을 처리하고, 이것이 종료되면 시스템 프로그램을 다시 불러 시스템 운영 제어권을 넘겨준다. IBM-PC상에서는 "System"의 입력으로 이 기능이 수행된다.

이와같이 사용하고자 하는 프로그램 형식(C컴파일러, 어셈블러, DBASE 등)을 사용자가 직접 불러 활용할 수 있다.

이러한 과정을 그림 4와 같이 기억소자 중심으로 설명해보자. 8086/286은 전원이 인가되면, 코드 세그먼트는 1로 세트되고 프로그램 카운터인 IP는 0으로 클리어되므로 시작번지는 "FFFF0" 번지가 되며 이 번지에 해당되는 기억소자는 반드시 ROM에 할당시킨다. 또한, 인터럽트 처리를 위한 벡터 주소가 0번지부터 3FF번지(256개의 인터럽트형 처리과정이 포함됨)이므로 이 영역도 ROM에 할당시켜야 한다.

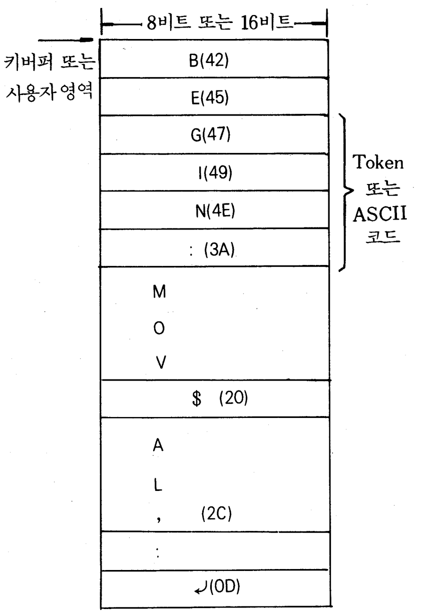
이와같이 지정된 기억소자를 영역별로 도시하면 그림 4와 같으며, 상위번지의 모니터 프로그램에서는 마이크로프로세서 자체의 주변 하드웨어를 제어하기 위한 초기화 프로그램이 사용되고, 어떤 프로그램을 사용할 것이냐에 따라 키입력장치로부터 프로그램을 지정하고 디스크로부터 이를 읽어 번역기 영역에 저장시킨다.

그림 4에서 실제 예를 들어 "A> MASM" 이 입력된 후 디스크로부터 부팅(Booting)된 상태를 보여주며(이의 세부적 내용은 어셈블러 구성에서 설명된다.) 이후부터는 MASM이라는 번역기가 본 컴퓨터 시스템을 관리하여 입력장치로부터 사용자 프로그램을 받아들여 소스프로그램을 저장영역에 기억시킨다.

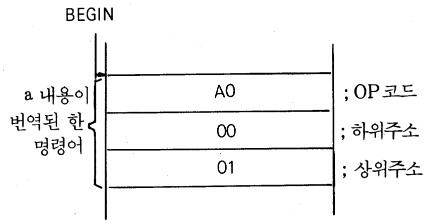
소스프로그램 영역에 모든 사용자 프로그램이 기억되면 이를 처리하게 되는데, 이때 MASM이라는 번역기가 작동하여 사용자 프로그램을 번역하여 기계어 프로그램으로 바꾸어 저장하였다가 처리한다. 이때 하단의 시스템 영역이란 키버퍼 등 MASM 번역기의 작동으로 특정한 기억영역을 의미하며, 현재 입력된 키값이 저장될 위치의 지정, 키버퍼의 내용이 소스프로그램에 이동되어 어느 위치에 저장될 것인지를 지정하는 등 사용자는 이용하지 못하는 영역을 말한다. 이 시스템영역의 워킹 메모리(Working Memory)에 관해서는 본 강좌 마지막회때 설명할 시스템 전체의 구성을 예로 들어 세부적으로 논하겠다.

그러면, 키보드로부터 한줄의 명령이 입력될 때 키버퍼에 어떻게 저장되

〔그림 5〕 키버퍼의 프로그램 저장내용

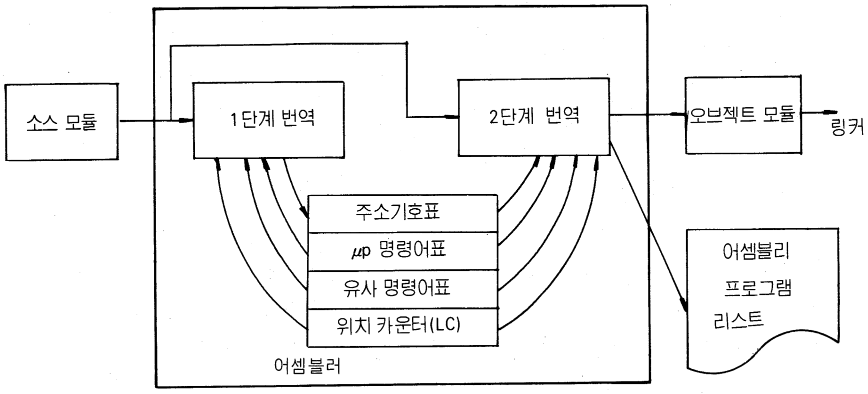


(a) 키버퍼의 값



(b) 기계어 명령어

〔그림 6〕 명령어 번역과정



며(이는 사용자 영역에서도 똑 같이 구성됨), MASM 번역기에 의하여 어떻게 처리되는가를 그림 5에서 보였다.

예를 들어 "BEGIN : MOV AL, TBL" 이란 명령어는 MASM 형식(다음절에서 설명됨)에 따라 작성된 프로그램(One Line)중의 한 부분이며, 이들은 1자당 1바이트 영역의 기억장소에 그림 5(a)와 같이 1 또는 0의 상태로 그 1자에 해당하는 ASCII 코드값이 저장된다. 이는 다시 MASM에 의해서 1줄씩 해당되는 기계어 명령어로 그림 5(b)와 같이 저장된다. 모든 프로그램이 이와 같은 과정을 통하여 번역된 후 마이크로프로세서에 의하여 각 명령어를 읽어(Fetch 동작), 수행(Execute 동작)한다.

어셈블러의 구성

컴퓨터와 사람이 대화하기 위한 언어로는 사람이 사용하는 언어와 흡사한 고급언어와 컴퓨터를 직접 제어할 수 있는 "1" 또는 "0"으로 구성된 기계어 등 여러 형태가 있으나 고급언어로 작성된 프로그램은 앞에서 언급했듯이 반드시 컴퓨터가 이해할 수 있는 1과 0의 신호형태로 변환되어야 한다. 따라서 본 절에서는 이들 번역기 구성에 의해 독자들의 이해를 돕기 위하여 비교적 간단하지만 모든 번역원리를 포함한 기호화된 프로그램의 번역기, 즉 어셈블러에 관하여 설명하겠다.

어떤 종류의 언어이든지 형식에 맞는 규격안이 있어야 하듯이 컴퓨터 언어에서도 각 언어에 따라 포맷(FORMAT) 이라는 고유형식이 존재한다. 따라서 본 절에서 고려할 어셈블리 언어에 대한 형식을 우선 정의해보면 다음과 같다.

```
<label>:<mnemonic><operand>;<comment>
```

여기서 〈label〉은 문자기호로 표시된 명령어가 기억되어 있는 주소번지로 최대 8자의 숫자 및 문자로 구성된다. 이들 8자는 어셈블러 구성자이며 임의로 정의할 수 있고, 번역기 구성상 기호표 구성에서 기억소자 할당에 따라 결정되고 필요하지 않을 경우에는 생략 가능하나 사용시에는 반드시 다음에 ":"이 뒤에 연결된다.

〈mnemonic〉은 명령어의 기능이 어떤 것인지를 정의한 것으로 마이크로프로세서가 사용될 수 있는 명령어를 그 의미에 따라 설정한 OP코드이다.

〈operand〉란 OP코드에서 적용될 정보값을 지시하는 것으로 정보값 또는 정보값이 들어있는 위치(이를 어드레싱 모드라 하며, 이미4월호에서 자세히 설명하였으므로 이를 참조하기 바람)를 가리킨다.

〈comment〉는 ";" 뒤에 사용되며 컴퓨터에는 전혀 의미가 전달되지 않고, 단지 컴퓨터 사용자들에게 이 명령어가 무엇인지를 설명하기 위한 설명문(주석문이라고도 함)이다. 이 부분도 필요없는 경우에는 생략할 수 있다.

이와 같은 형식에 따라 명령어가 구성되어야 한다. 어셈블리어의 구성방법은 사용 형태에 따라 기계어 명령어로 번역이 되어 컴퓨터를 구동시키는 실처리 명령어와 단지 프로그램 번역기에 정보를 제공해 주는 유사명령어로 구성된다(그림 6 참조).

기계어 구동을 위한 명령어는 지난달에 이미 기술하였으므로 참고하기 바라며, 여기서는 유사명령어에 관해 기본적으로 필요한 것만 설명하겠다.

먼저, 기억소자 지정을 위한 명령어로서 사용변수를 바이트(8비트), 워드(16비트)로 지정하는 DB(Define Byte),

DW(Define Word)가 있고 프로그램의 시작과 끝을 번역기에 알려주기 위한 ORG 및 END문, 세그먼트 지정을 위한 명령어, 변수값 지정을 위한 명령어, 화일정보 및 비트지정을 위한 명령어 등 어셈블러 구성에 따라 다양하게 이루어졌다.

이와같이 유사명령어는 프로그램 번역을 위한 정보를 제공하는 것으로 응용을 위해서는 "MASM"이라는 기존에 널리 사용되고 있는 86-매크로어셈블러를 각자 습득하기 바란다.

어셈블리어로 구성된 예제 프로그램을 **표 2**에 설명하였다. 기계어로 번역할 때 명령어에 사용되는 변수 때문에 한번에 읽어 번역할 수 없기 때문에 2번에 걸쳐 읽고 처리하여야 하는데 이를 투패스(Two Pass) 어셈블러라 한다.

즉 첫번째 읽을 때는 레이블을 읽어 주소표를 작성하는 일을 하고, 2번째 과정에서는 1과정에서 생성된 주소표와 명령어표를 활용하여 그림 6에 보여준 것과 같이 각 명령어를 기계어로 번역하는 일을 한다.

1단계 번역과정은 **그림 7**과 같이 한줄씩 읽어 레이블이 사용된 경우를 찾아 기호 주소표를 작성하는 과정이다. 2단계 번역과정은 **그림 8**과 같이 어셈블러에 기억된 OP코드표와 유사명령어표 및 1단계 번역 과정에서 만들어진 기호 주소표를 사용(그림 4의 어셈블러 기억 영역표 참고)하여 번역한다. 여기서 한줄을 읽는다고 함은 그림 5에서와 같이 리턴 (ASCII 코드로 0DH) 표를 확인하는 것까지 1줄에 해당되며 해당 명령어를 효율적으로 찾기 위해서 각 명령어표를 **표 3**과 같이 정의하게 된다(독자의 이해를 쉽게하기 위하여 간편하게 구성하였음).

표 3(a)의 명령어표는 매명령어마다 명령어 기호표(7바이트)와 이의 기계어 등가표 (2바이트) 및 명령어 형식(1바이트) 등을 10바이트로 구성한다. 따라서 300여개의 명령어라면 이들 명령어표는 3000바이트로 구성되며, 번역하고자 할 때 사용한 명령어는 이들 3KB 이내에 있으므로 이 3KB영역내에서 찾아 오브젝트 프로그램 기억위치에 기계어 코드로 기억시키면 된다.

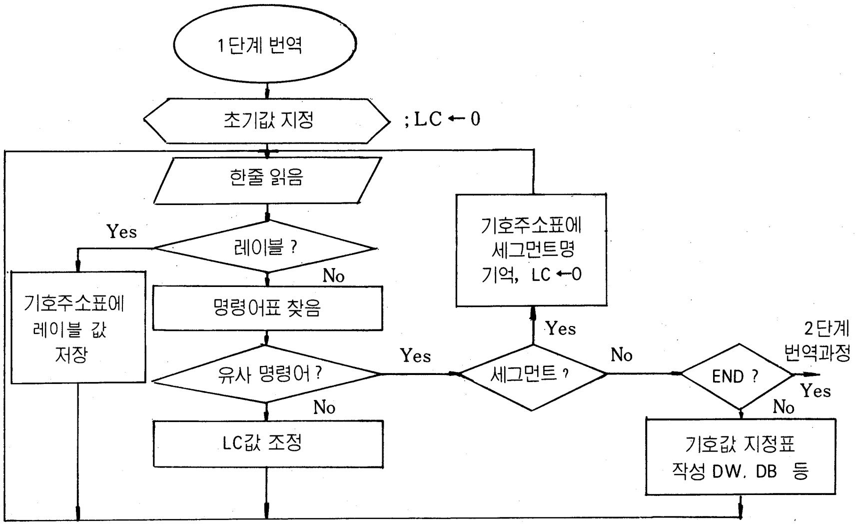
기호 주소표는 1단계 번역과정에서 표 3(c)와 같이 생성되며, 명령어에 따라 기호 주소표를 찾아야 하는가가 결정되어 사용 변수명을 그 값으로 대치하게 된다. 유사명령어는 기계어로 번역되는 것이 아니고 즉시 처리하기 위한 제어 명령어이므로 12바이트씩 표 3(b)와 같이 할당하여 사용기호를 찾아 이를 처리할 수행번지로 보내 이를 처리하는 것이다.

지면상 세부적인 설명은 생략하기로 하고 독자 각자가 이들 프로그램을 작성하여 구성해 보길 바란다. PC

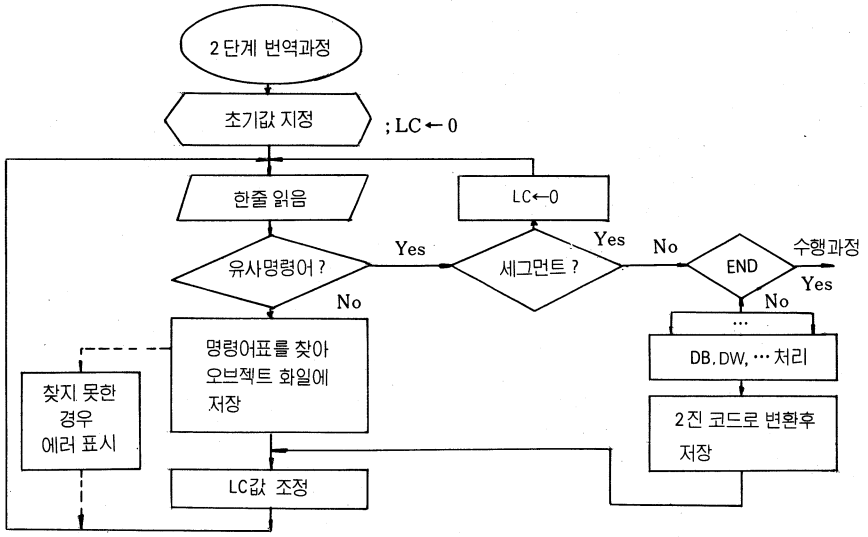
〔표 2〕 예제 프로그램

소스 프로그램	기계어 프로그램 위치주소	기계어 프로그램 명령어 코드
DATA_SEG SEGMENT		
OPR1 DW 12	0000	0C00
OPR2 DW 230	0002	E600
RSLT DW ?	0004	????
DATA_SEG ENDS		
CODE_SEG SEGMENT		
ASSUME CS:CODE_SEG, DS: DATA_SEG		
START: MOV AX, DATA_SEG	0000	B8……
MOV DS, AX	0003	8ED8
MOV AX, OPR1	0005	A10000
ADD AX, OPR2	0008	03060200
JGE STORE	000C	7D02
NEG AX	000E	F7D8
STORE: MOV RSLT, AX	0010	A30400
HLT	0013	F4
CODE_SEG ENDS		
END START		

〔그림 7〕 1단계 번역과정



〔그림 8〕 2단계 번역과정



〔표 3〕 명령어표의 구성

(각 명령어당 10바이트로 구성됨)

기호화된 명령어(7바이트)	명령어 형(1바이트)	2진 OP 코드값(2바이트)
ADD ……	0 0 0 0 ××××	OP코드 MOD/RG
⋮	⋮	⋮
MOV ……		
⋮		
OUT ………		

(a) 명령어표 구성

(각 명령어당 12바이트)

유사명령어(8바이트)	명령어 처리를 위한 시작번지(4바이트)
ASSUME	×××× ××××
⋮	
DEFINE	코드세그먼트 값 실효주소 값
⋮	⋮
END	
⋮	

(b) 유사 명령어표 구성

(각 기호당 10바이트)

기호(8바이트)	기호값(2바이트)
BEGIN	0 0 0 0
⋮	⋮
COUNT	0 1 0 0

(c) 기호 주소표 구성

레지스터와 프로세서

김정선 / 한국항공대학 전자과 교수

레지스터 전송

●개요

디지탈시스템은 게이트와 플립플롭들로 구성된 순서논리 시스템이다. 순서논리 회로는 상태표를 사용하여 설명할 수 있으나 대형 디지탈시스템을 설명하는데 있어서 상태표를 사용하여 구체적으로 설명하기에는 게이트수가 너무 많기 때문에 대단히 어려운 문제이다.

이와 같은 어려운 점을 해결하기 위하여 일반적으로 디지탈시스템은 모듈접근법(Modular Approach)으로 설계한다. 즉 전시스템을 각기 어떤 기능을 수행하는 모듈화된 부시스템(Modular Subsystem)으로 분할하는 것이다. 이들 각 모듈은 레지스터, 카운터, 멀티플렉서, 연산소자, 논리회로 등의 기능을 수행하는 장치들로 이루어진다. 이와같은 모듈들은 공통적인 데이터나 제어회로들로 상호 연결되어서 디지탈 컴퓨터와 같은 시스템을 형성하게 된다. 전형적인 디지탈 시스템 모듈로는 디지탈 컴퓨터의 처리장치를 들 수 있다.

디지탈 시스템의 모듈을 형성하기 위해서는 계수적 기능을 수행하는 장치들을 상호 연결하게 되는데 이는 순서논리나 조합논리 설명기법으로 설명하기 어렵고 게이트나 플립플롭차원에서 디지탈시스템을 설명하는데 적합하도록 개발된 기법이다. 레지스터나 디코더 등의 차원에서는 레지스터 전송논리 방법을 적용하는 것이 개발되어 있으므로 디지탈시스템의 기본 구성요소가 게이트나 플립플롭이 아니고 레지스터가 된다. 이 방법으로 정보의 흐름이나 저장된 데이터들의 처리들을 표현할 수 있게 된다.

레지스터 전송논리 방법은 수식들과 프로그래밍 언어에서의 문장들이 서로 유사한 것을 사용한다.

디지탈시스템을 기술하는 레지스터 전송논리 방법으로는

① 시스템내의 한개의 레지스터와 그들의 기능
② 레지스터에 저장된 2진 코드화된 정보
③ 레지스터에 저장된 정보에 대하여 행해지는 작동
④ 일련의 작동순서를 시동시키는 제어함수

등의 4가지를 들 수 있다.

레지스터에 저장된 2진 정보는 2진수나 2진 코드화 10진수, 영문숫자(Alphanumeric), 제어정보 등 2진 코드화된 모든 정보를 말한다.

레지스터에 저장된 데이터들에 관하여 행해지는 조작을 마이크로 작동(Micro Operation)이라고 한다. 마이크로 작동은 클럭펄스 1주기동안에 병렬로 수행되는 기본작동이다. 그 결과는 전에 있던 정보와 대체되거나 다른 레지스터로 옮겨진다.

예를 들어 자리이동(비트이동), 계수(Count), 덧셈, 소거, 로드(Load) 등을 들 수 있다.

마이크로 작동은 다음과 같이 대별된다.

① 레지스터간의 전송
② 연산 마이크로 동작
③ 논리 마이크로 동작
④ 자리이동 마이크로 동작

●레지스터간의 정보전송

레지스터의 구성은 플립플롭 1개로 이루어져 1비트의 2진 정보로 입출력을 제어하는 조합논리회로가 다수개 모인 것이다. 즉 여러개의 플립플롭과 게이트들로 구성되는데 플립플롭은 2진 정보를 저장하고 게이트들이 레지스터에 2진정보가 어떻게 전송되는지를 제어한다. 그리고 8비트와 16비트가 주로 이용된다. 레지스터는 **그림 1**과 같이 표현된다.

일반적으로 디지탈시스템은 많은 레지스터들을 필요로 하고 있으며 이들 레지스터간에 데이터를 전송하기 위하여는 각각의 전송로가 구성되어야 한다.

그림 2에서와 같이 3개의 레지스터에는 6개의 전송로가 필요하며 만일 각 레지스터가 n비트로 구성되어 있다면 6n개의 배선이 각각 필요하다. 레지스터의 수와 비트수가 클경우 이와같은 각각의 전송로 구성방식에 의한 회로구성은 회로의 복잡성 등에 의한 회로설계의 어려움, 단가의 상승 등의 문제가 따른다.

그러나 이러한 문제점들은 **그림 3**과 같은 구조의 전송로방식을 채택하므로서 해결된다. 즉 그림 3은 공동의 전송로(BUS)를 통하여 적당한 시차를 두고 모든 데이터를 이동시킬 수 있도록

하는 구조이다. 이때 공동의 전송로를 버스(BUS)라 하고 이러한 구조의 시스템을 버스시스템이라 한다.

예를 들어 이러한 시스템에서 R_1의 데이터를 R_2로 옮기고자 한다면 S_1과 S_5만을 ON으로 하고 나머지는 OFF로 하므로서 R_1의 출력은 버스를 통하여 R_2의 입력에 가해지므로 R_1의 데이터를 R_2로 이동시킬 수 있게 된다. 물론 이때 또다른 데이터의 이동은 동시에 이루어질 수 없으며 각각의 전송로를 가진 시스템에 비하여 시간적인 지연은 크다. 그러나 버스라인은 레지스터의 비트수에 대응하게 되어 레지스터의 수가 많든 작든 버스라인은 비트수만큼만 필요하게 되며 회로구성이 간단하게 된다.

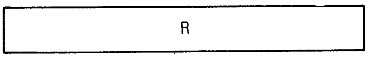
어떤 시스템에서는 수개의 레지스터와 더 많은 정보를 동시에 기억시켜 두고 처리하기 위하여 메모리를 필요로 하며 메모리의 정보를 레지스터로 혹은 레지스터의 정보를 메모리로 이동시킨다. 그리고 메모리의 번지지정을 위한 번지정보(Address Data)를 역시 레지스터에 저장하였다가 필요시 메모리의 번지버스(Address Bus)에 전송하므로서 필요한 번지의 메모리를 선택하도록 할 수 있다.

그림 4에 메모리와 레지스터 간에 정보를 교환할 수 있는 구성도를 보였다. 이 그림에서 메모리의 번지는 번지용 버스로부터 얻어지며, 4개의 번지정보 레지스터(A_0~A_3)들이 이 버스에 멀티플렉서에 의하여 선택 연결된다. 메모리로부터의 출력은 4개의 레지스터 중에서 행선 결정용 디코더(Destination Decoder)에 의하여 선택되어진 레지스터로 출력버스를 통하여 전송된다. 레지스터(B_0~B_3)의 출력은 멀티플렉서(MUX2)에 의하여 B_0~B_3중 하나가 선택되며 메모리 입력버스에 접속된다. 그리고 멀티플렉서(MUX1)에 의하여 선택된 번지 레지스터의 데이터에 의하여 선택된 메모리에 전송된다.

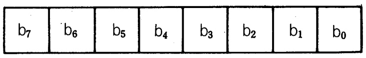
●간단한 구조의 컴퓨터

컴퓨터의 구조와 동작원리를 이해하기 위하여 그림 5와 같은 간단한 구조의 컴퓨터 구성예를 들고 살펴보기로 한다. 그림 5에서 각 구성요소들은 PC(Program Counter), MAR(Memory Address Register), IR(Instruction Register), IDC(Instruction Decoder & Controller), TD(Timming Decoder), T(Clock Generater), 메모리, MBR(Memory Buffer Register), A(Register A), B(Register B)와 조합논리회로들로 구성된다.

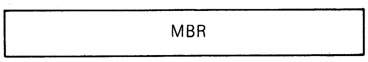
〔그림 1〕 레지스터의 블록표시



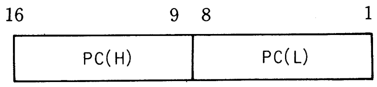
(a) 레지스터 R



(b) 개개의 비트소자 표시

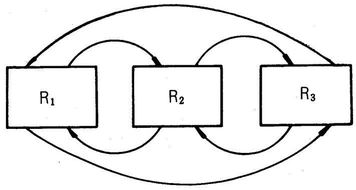


(c) 명칭 및 번호첨부 표시

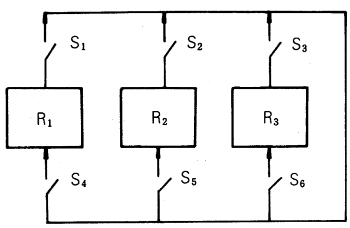


(d) 레지스터의 부분표시

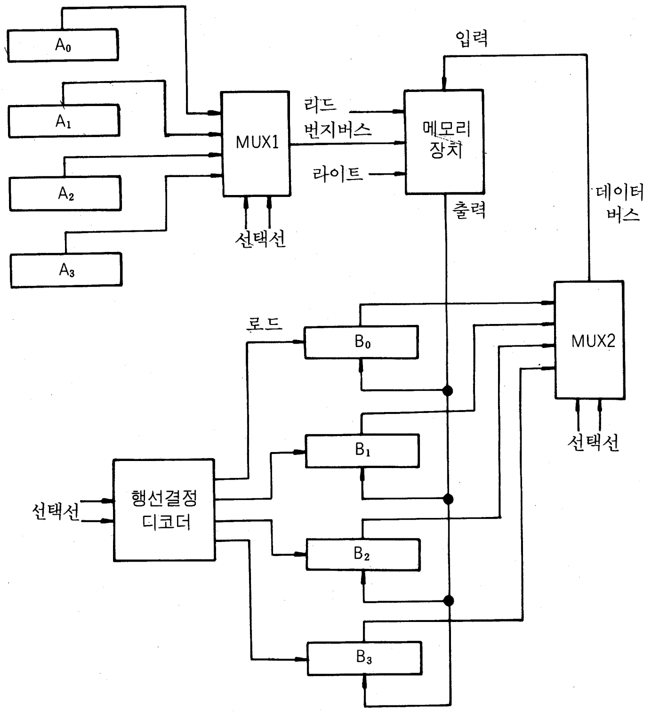
〔그림 2〕 레지스터간의 정보전송



〔그림 3〕 버스형 전송로 개념도



〔그림 4〕 레지스터와 메모리의 정보전송을 위한 회로 구성도



이를 각 구성요소들에 대해 알아보면 다음과 같다.

① PC : 하나씩 증가해서 연속적으로 메모리에 저장된 명령어들을 읽어낼 수 있도록 한다. 그리고 PC는 언제나 다음에 수행될 메모리내의 명령어번지를 담고 있다.

② MAR : 메모리의 내용을 읽거나 메모리에 새로운 데이터를 써넣기 위한 메모리 번지지정 데이터를 저장한다.

③ IR : 메모리에 저장되어 있는 명령어들중 PC에 의하여 지정되어 수행하여야 할 명령어를 일시 저장하였다가 명령어 해독 및 제어회로(IDC)에 전송한다.

④ T와 TD : T에 의하여 발생된 클럭신호를 디코딩하여 t_0~t_5중 하나를 선택하여 순차적으로 출력시킨다.

⑤ 메모리 : 256×8 즉 8비트(1바이트) 메모리가 256개로 구성되는 메모리 블록으로 어드레스선(번지 지정선)을 8개 필요로 한다.

⑥ MBR : 메모리의 입출력 데이터를 일시 저장용으로 사용한다.

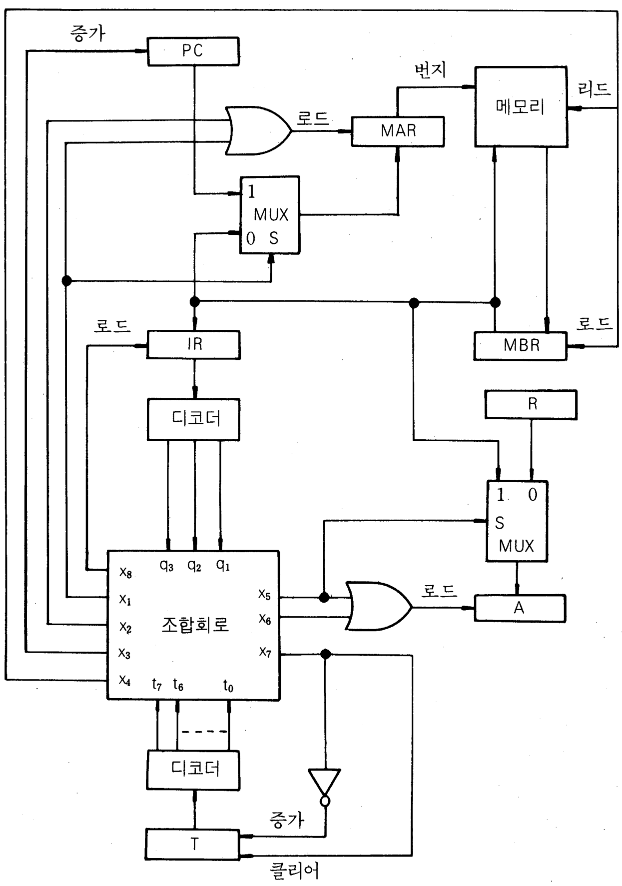
⑦ 레지스터 A, B : 데이터의 일시저장용으로 더 복잡한 구조의 컴퓨터에서 연산이나 기타 다른 동작을 위한 레지스터나 기타 회로를 대표하는 레지스터로 생각한다.

⑧ 명령어의 디코더와 TD의 출력 : 조합논리회로를 통하여 회로에서와 같이 각 회로요소를 동작할 수 있게 한다.

컴퓨터가 프로그램에 의하여 어떠한 처리를 할때 메모리에 차례로 저장되어 있는 프로그램중 명령코드를 읽어 명령어 디코딩을 위한 일시저장용 레지스터에 저장시키고, 이 레지스터의 출력이 명령어 디코더에 의하여 디코딩되어 명령어에 따른 다음 처리를 한다. 이때 명령어가 메모리로부터 명령어 레지스터에 전송되기까지를 패치사이클(Fetch Cycle)이라하며 명령어에 의한 처리가 완료될때 까지를 수행사이클(Excute Cycle)이라 한다.

메모리에 저장된 프로그램에는 명령

〔그림 5〕 간단한 컴퓨터 구성 예



코드와 명령코드에 따라 그 명령을 수행하는데 필요한 데이터나 메모리 번지지정용 데이터가 명령어에 수반되어 저장된다. 한 명령어에 수반되는 데이터나 번지지정용 데이터를 오퍼랜드라 하며 한 명령세트는 명령어+오퍼랜드로 구성된다.

전술한바와 같이 데이터의 전송이나 레지스터의 내용변경의 한동작 즉 클럭신호 한주기동안의 작동을 마이크로 작동이라 하며, 최소한의 기능동작 즉 한 명령세트의 동작이 완료되기까지의 동작을 매크로동작이라 한다.

이 매크로 동작의 한 예로 오퍼랜드가 메모리 번지데이터인 명령 "LDA ADRS"에 의해 ADRS번지의 메모리 내용을 레지스터 A로 옮기는 매크로 명령의 마이크로 작동은

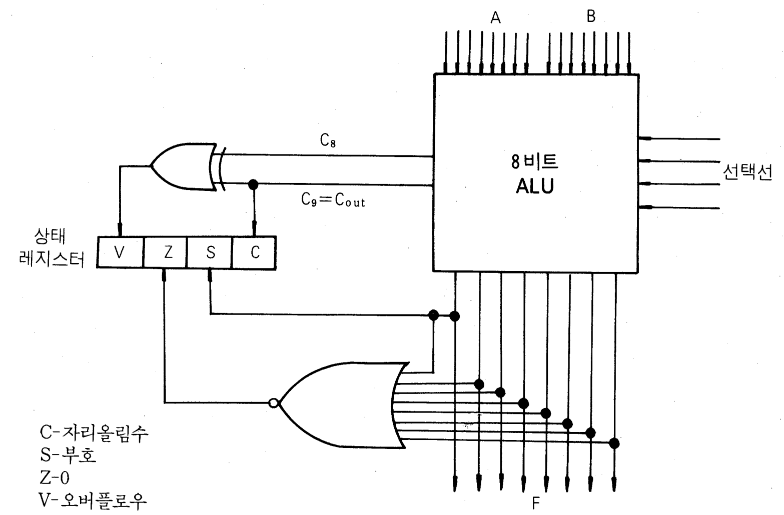
g_3t_3 : MAR←PC (다음 명령어 번지를 전송)
g_3t_4 : MBR→M, PC←PC+1 (ARD를 읽고 PC를 증가)
g_3t_5 : MAR→MBR (오퍼랜드의 번지를 전송)
g_3t_6 : MBR←M (오퍼랜드를 읽음)
g_3t_7 : A←MBR, T=0 (오퍼랜드를 A로 보내고 페치사이클로 돌아감)

과 같이 컴퓨터 하드웨어식 표현을 할 수 있다.

이러한 표현방법으로 간단한 컴퓨터의 레지스터전송을 위한 컴퓨터 제어함수와 마이크로 작동들을 표 1에 요약해놓았다.

표 1에서 $t_0t_1t_2$는 작동코드(OP코드)를 IR로 읽어내는 명령어 페치사이클이다. 즉 T와 TD가 클리어되어 t_0부터 출력이 나오고 다음 클럭에서 t_1, 세번째 클럭에서 t_2의 출력이 나오며 이 기

〔그림 6〕 4비트 상태레지스터를 가진 8비트 ALU의 블록도



〔표 1〕 간단한 컴퓨터 제어함수와 마이크로 작동

페치	t_0 :	MAR ← PC
	t_1 :	MBR ← M, PC ← PC+1
	t_2 :	IR ← MBR
MOV	q_1t_3 :	A ← R, T ← 0
LDI	q_2t_3 :	MAR ← PC
	q_2t_4 :	MBR ← M, PC ← PC+1
	q_2t_5 :	A ← MBR, T ← 0
LDA	q_3t_3 :	MAR ← PC
	q_3t_4 :	MBR ← M, PC ← PC+1
	q_3t_5 :	MBR ← MBR
	q_3t_6 :	MBR ← M
	q_3t_7 :	A ← MBR, T ← 0

간동안에는 무조건 메모리로 부터 IR로 정보가 이동되며 이 정보는 명령어 디코더에 가해지므로 명령어코드로 간주된다. t_3동안 일어나는 마이크로 동작은 IR로 페치된 OP코드에 따라서 각기 다른 q_1~q_3의 조합출력이 나타난다. 이들 출력 q_1~q_3와 순차적으로 나타나는 t_3~t_7이 조합논리회로를 통하여 제어출력 x_1~x_8을 출력시키므로서 필요한 부분들이 순차적으로 동작하게 된다.

프로세서의 논리와 설계

●프로세서의 구성

컴퓨터나 디지탈시스템에서 처리장치(Processor Unit)는 디지탈시스템의 모든 작동들을 집행하는 부분이다. 구성은 여러개의 레지스터들과 산술(Arithmetic), 논리(Logic), 자리이동(Shift)과 전송(Transfer), 마이크로 작동(Microoperation)들을 집행하는 디지탈기능들로 되어 있다. 마이크로 작동 순서를 관장하는 제어장치와 결합된 처리장치를 중앙처리장치(Central Processor Unit : CPU)라 한다.

처리장치내의 많은 레지스터들은 대부분 버스형태로 연결하거나 처리시간이 매우 빠른 작은 용량의 메모리형태로 배열하게 된다. 레지스터들은 데이터의 전송 뿐만 아니라 여러가지 마이크로 작동을 위해 상호간 통신을 한다.

연산장치

프로세서내에서 처리할 데이터의 변형처리를 하는 곳이 연산장치와 시프터이나 여기서는 연산장치에 대해서만 알아본다. 연산장치는 산술연산장치와 논리연산장치 그리고 산술연산 결과 얻어지는 각종 플래그(Flag) 즉 오버플로우(Overflow), 캐리(Carry), 부호(Negative) 등의 상태 레지스터들의 집합이다. 상태레지스터와 산술연산회로와의 연결관계에 대한 간단한 예를 그림 6에 보였다.

여기서 C, S, Z, V에 대해 알아보자.

① 상태 비트 C ; ALU의 캐리출력이 1이면 세트(1로)되고 0이면 클리어(0로)된다.
② 비트 S ; ALU의 최상위 비트(부호비트라고도 함)가 1이면 세트되고 0이면 클리어된다.
③ Z비트 ; ALU의 출력결과 모든 비트가 0이면 세트되고 어느 한 비트라도 0이 아니면 클리어된다.
④ V비트 ; ALU의 캐리 C_8과 C_9의 Exclusive-OR에 의하여 세트/리세트된다. 이것은 수들이 모듈로 2의 보수법으로 표시되었을때 오버플로우 조건이다. 8비트 ALU에 대해 V는 결과가 127보다 크거나 −128보다 작으면 세트된다.

여기서 상태비트는 ALU 연산 직후에 두값 A와 B간의 관계를 알기 위하여 필요하다.

논리연산회로는 그림 7에 나타낸 것과 같은 것을 생각할 수 있으며 그림 7에서 알 수 있는 것과 같이 A_iB_i에 대한 논리연산결과를 F_i에서 얻을 수 있다.

그리고 앞에서의 산술회로와 이 회로를 결합하였을 경우 그림 8의 산술/논리연산의 조합회로가 구성된다.

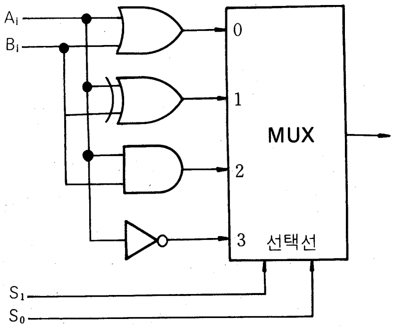
처리장치의 구성과 명령코드의 개념

상술한 구성요소들을 조합하여 프로세서를 구성하였을 경우 각 구성요소들에 가해지는 제어함수들의 적절한 조합으로 임의의 필요한 마이크로 작동을 실현할 수가 있다. 즉 처리장치의

〔표 2〕 그림 9의 제어변수 기능

2진코드	선택 변수들의 기능					
	A	*B*	*D*	*F* with $C_{in}=0$	*F* with $C_{in}=1$	*H*
0 0 0	입력데이터	입력데이터	없음	*A*, *C*←0	*A*+1	자리 이동 안함
0 0 1	*R*1	*R*1	*R*1	*A*+*B*	*A*+*B*+1	오른쪽 자리 이동, $I_R=0$
0 1 0	*R*2	*R*2	*R*2	*A*−*B*−1	*A*∸*B*	왼쪽 자리 이동, $I_L=0$
0 1 1	*R*3	*R*3	*R*3	*A*−1	*A*, *C*←1	출력 bus에 0을
1 0 0	*R*4	*R*4	*R*4	A∨*B* ⊕*B*	—	—
1 0 1	*R*5	*R*5	*R*5	*A*, C←	—	carry가 있는 환상 오른쪽 자리 이동
1 1 0	*R*6	*R*6	*R*6	*A*∧*B*	—	carry가 있는 환상 왼쪽 자리 이동
1 1 1	*R*7	*R*7	*R*7	*A*	—	—

〔그림 7〕 1비트 논리연산회로

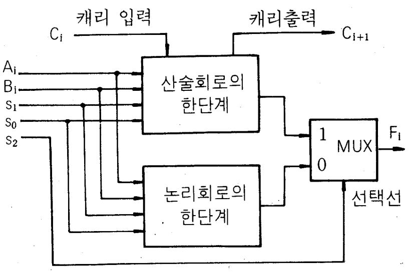


(a) 논리도

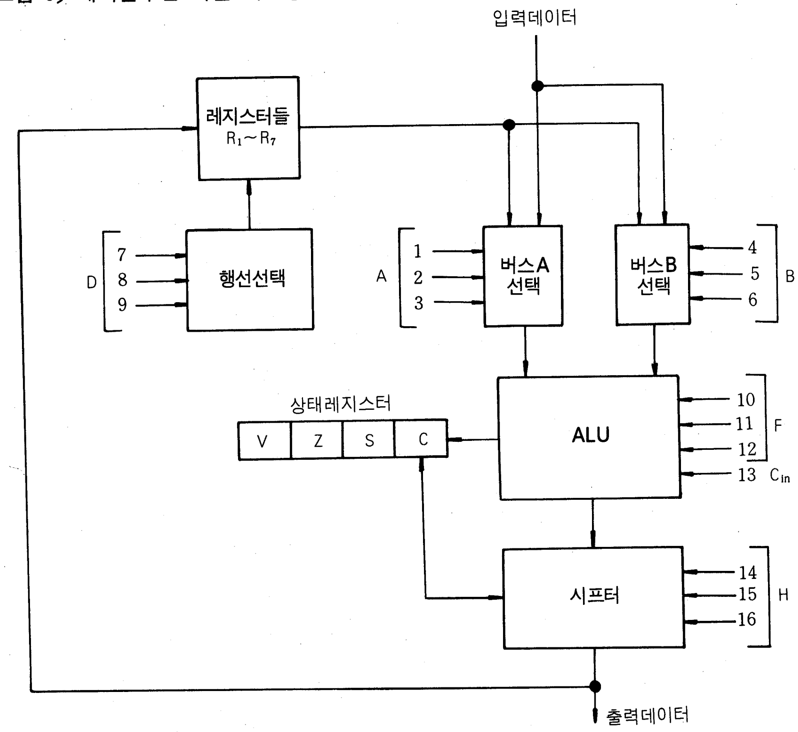
S_1 S_0	출 력	연 산
0 0	$F_i=A_i+B_i$	OR
0 1	$F_i=A_i\oplus B_i$	XOR
1 0	$F_i=A_iB_i$	AND
1 1	$F_i=A_i'$	NOT

(b) 기능표

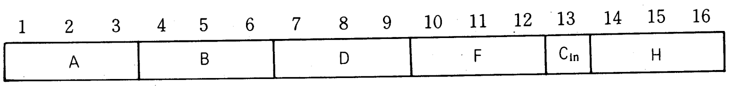
〔그림 8〕 논리회로와 산술회로의 결합



〔그림 9〕 제어변수를 가진 처리장치



(a) 블록도



(b) 제어 워드

선택 변수들(제어함수들)은 어떤 주어진 클럭펄스동안 장치안에서 수행될 마이크로 작동들을 제어한다.

그림 9는 상술한 구성요소 들로 조합 구성시킨 처리장치의 예이다. 여기에는 R_1~R_7의 레지스터 7개와 1개의 상태레지스터가 있다. 7개의 레지스터는 그 출력이 멀티플렉서(그림에서 버스 A 선택과 버스 B 선택)에 의하여 선택되어 ALU의 한 입력으로 연결된다.

이때 버스 선택 A나 B가 선택제어선(A의 1, 2, 3, B의 4, 5, 6)을 3개씩 가지고 있으므로 각 선의 1, 0의 상태조합은 $2^3=8$로 8개를 선택할 수 있다. 즉 레지스터 7개와 입력데이터를 포함하여 A, B에서 각기 하나를 선택할 수

있다.

ALU의 출력은 시프터를 통하여 외부 출력단자와 D(7, 8, 9)의 행선 선택에 의해 선택된 레지스터로 저장될 수 있다. 이 장치에서 선택 변수는 A에 3개(1, 2, 3), B에 3개(4, 5, 6), D에 3개(7, 8, 9) 등 모두 16개이고 이들 기능은 전술한 각 모듈(각 구성요소들)의 제어기능에 비추어 보았을때 그의 조합 기능은 표 2와 같다.

그림 9와 표 2에 의해 뺄셈의 마이크로 동작 예로서

$R_1 \leftarrow R_1 - R_2$; R_1에서 R_2를 빼서 R_1에 저장

이 가능하도록 하기 위한 제어 입력은

A	B	D	F	C_{in}	H
001	010	001	010	1	000

이어야 하는데 이를 모았을 경우 001 01000101011000 의 하나의 제어용 워드(Word)를 유도해 낼 수 있다. 이는 우리가 2진수 형태로 읽기 힘드므로 4비트씩 분리하여 16진수로 표기 하면

0010,	1000,	1010,	1000
2	8	A	8

즉 $(28A8)_{16}$로 표기할 수 있고, 8비트 1바이트 단위로 표기한다면 2바이트로 표현된다.

이러한 제어용워드의 형태언어를 기계어라 한다.

이 시스템에서 제어선이 16개이므로 16개의 조합으로 얻을 수 있는 기능의 수는 2^{16}가지가 되지만 중복기능 혹은 불필요한 기능들을 제외하고 꼭 필요한 기능들 만을 발췌하여 실행시키도록 한다면 PLA회로나 ROM, 혹은 디코더 등을 이용하여 최종 제어입력선 수를 줄일 수 있다. 혹은 일련의 연속적인 마이크로 동작을 시킬 수 있는 마이크로 프로그램 등에 의하여 필요한 명령의 수를 줄일 수도 있다. 특히 각 모듈(Module) 즉 ALU나 시프터 등의 구체적인 구성은 사용목적에 따라 적절한 형태의 구성을 하여야 할 것이다. PC

MCS-51 트레이닝 보드

이상민/광운대 전자계산기공학과 C그룹

제작에 앞서

현추세가 8비트 프로세서의 영역이 16비트 프로세서의 강세에 눌려 점점 축소되고는 있지만 8비트 프로세서가 사라질 것이라고 생각하지는 않는다. 그 이유는 8비트 프로세서로 시스템을 제작할 경우 16비트 프로세서보다 간단해진다는 장점 때문이다.

여기서 소개할 원칩 마이컴은 이러한 8비트 프로세서의 장점을 이용하여 기존의 범용 CPU보다 시스템 구성을 간단하게 한 것이다.

원칩 마이컴이라는 것은 말그대로 하나의 칩안에 시스템 구성에 필요한 여러가지 부품을 넣어 놓은 것으로서 이는 원칩 마이크로 컴퓨터라고도 한다. 이 칩은 컴퓨터에서 가장 기본적인 마이크로프로세서의 역할과 데이터의 입·출력과정, 메모리의 사용법과 구조 등을 쉽게 배울 수 있도록 구성되어 있다.

본 트레이닝 키트의 제작에는 시중에서 접하기 쉬운 Z80이나 6502 CPU로도 시스템을 구성할 수 있으나 MCS-51 패밀리를 사용하였다. 이 MCS-51 패밀리는 CPU에서 동작할 프로그램이 들어 있는 롬과 이 프로그램에서 필요한 작은 용량의 램, 시리얼통신을 위한 포트, 타이머 카운터 등을 내장하고 있고 ADC(AD Converter)나 DAC(DA Converter) 등과 연결하여 사용할 수 있다.

이와 같이 MCS-51 패밀리는 롬과 램을 내장하고 있어 제어가 가능하다는 장점을 갖고 있다. 이 MCS-51 패밀리의 기능을 좀더 자세히 알아보면 다음과 같다.

① 4K 바이트 롬(8051칩에서만 가능)
② 128 바이트 램
③ 32개의 I/O 라인(4개의 8비트 포트)
④ 2개의 16비트 타이머/카운터
⑤ 프로그램가능한 양방향성 직렬채널
⑥ 128K 메모리 확장 가능

MCS-51 패밀리

MCS-51 패밀리의 종류는 표 1과 같다. 여기서 3개의 CPU는 모두 같은 명령어를 가지고 있다. 각각의 다른점은 롬의 유무와 그 롬의 종류가 무엇인가 하는 것이다.

우선 8051AH는 롬이 내장되어 있기는 하나 이 롬에 프로그램을 자유로이 기록할 수 없는 타입이다. 따라서 CPU를 제조하는 과정에서 프로그램을 기록해야 하기 때문에 이를 MASK-PROM 이라고 한다. 반면에 8051H는 EPROM이 내장되어 있어 사용자가 임의로 프로그램을 기록할 수 있는 타입이다. 그리고 8031AH는 롬을 가지고 있지 않다.

이 3가지 CPU중 8031AH를 제작에 사용하였다. 이 CPU를 사용한 이유는 8051H의 경우 시중에서 구하기 어려운 원칩 프로그래머를 반드시 사용해야 하는데 비해 8031AH는 우리가 쉽게 접할 수 있는 롬 프로그래머를 사용하기 때문이다.

그리고 8031AH는 일반적인 범용 CPU와는 다른 메모리구조를 가지고 있다. 그림 1은 8031AH의 메모리 맵을 나타낸 것으로, 특이한 점은 프로그램 메모리와 데이터 메모리가 분리되어 있다는 것이다. 일반적으로 트레이닝 보드는 프로그램을 기계어로 직접 메모리에 기록시키고 그 주소를 부름으로써 자신의 프로그램을 실행시킨다.

다음은 8031AH와 프로그램 메모리와의 데이터 전송에 관계하는 명령어의 전부이다.

```
MOVE A,@A + DPTR
MOVC A,@A + PC
```

어셈블리언어에 능통한 독자라면 이 명령어를 보고 벌써 알았을 것이다. 이 8031AH는 프로그램 메모리에서 CPU로 데이터를 로드하는 명령어밖에 없

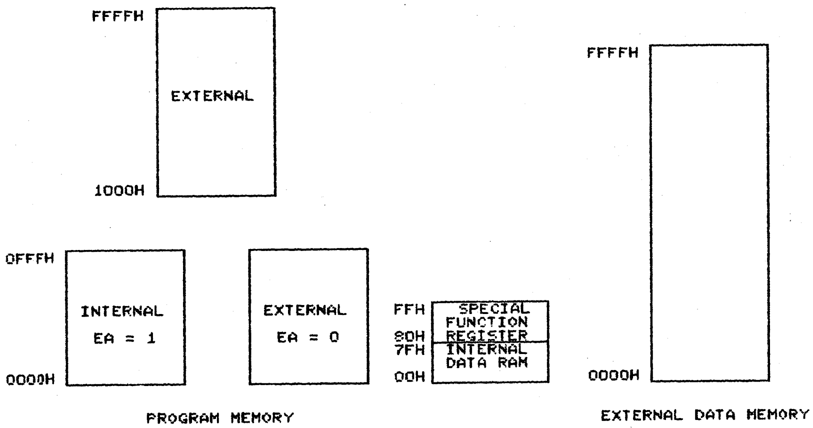
〔표 1〕 MCS-51 패밀리

종 류	프로그램 메모리 용량	데이터 메모리 용량
8051 AH	4K ROM	128 바이트
8031 AH	NONE	128 바이트
8051 H	4K EPROM	128 바이트
80C51	4K ROM	128 바이트
80C31	NONE	128 바이트

다. 따라서 프로그램을 프로그램 메모리에 기입할 수 없다. 이 문제를 해결하기 위해서 취한 방법이 실행될 프로그램을 데이터 메모리에 써넣고 그 메모리 주소를 프로그램 메모리의 위치로 이동시키는 방법이다.

또한 8031AH에 대해 알아두어야 할 것은 I/O를 메모리영역에 포함시키는 Memory Mapped I/O방식을 사용해야 하기 때문에 여기서 사용하게 될 8255와 273(D 플립플롭)은 데이터 메모리영역에 할당되어 있다는 점이다.

〔그림 1〕 8031AH의 메모리맵



제작과 회로

이쯤해서 8031AH에 대한 소개는 끝내고 본론으로 들어가 보자. 먼저 제작할 시스템에 대하여 알아보면, 이 시스템은 현재의 상태를 확인할 수 있는 표시장치가 필요하고, 메모리의 주소를 지정하고 그 데이터를 입력할 수 있는 입력장치(Key Pad)가 필요하다. 이 시스템의 전체 회로도는 그림 2 와

〔그림 2〕 전체회로도

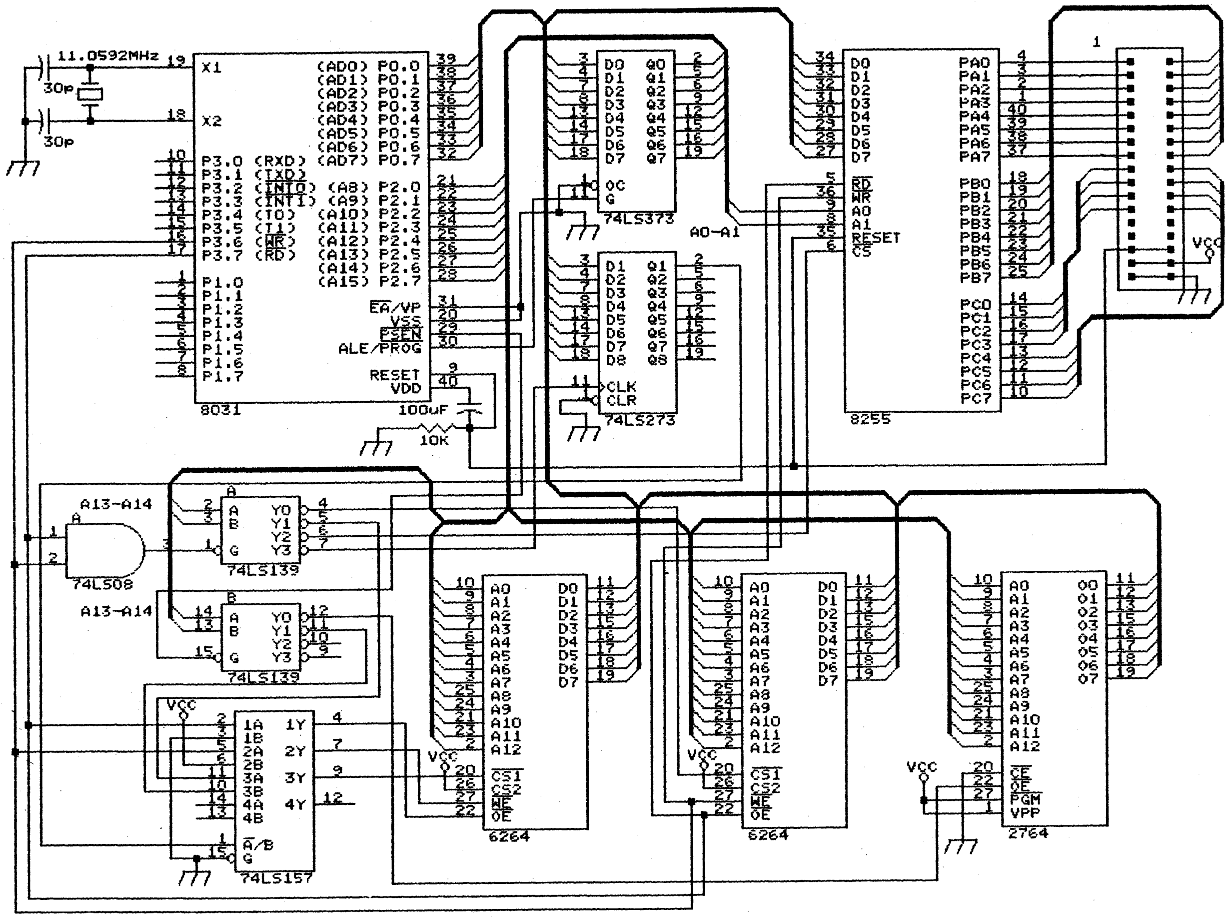


그림 3에서 보여준다.

●회로구성

이제 이 회로의 구성에 대해서 알아보기로 한다. 먼저 그림 4에 나타낸 8031AH 의 3개의 타이밍도를 보면서 CPU와 메모리간의 인터페이스와 메모리 맵에 대해 알아보자. 8031AH 4개의 포트중에는 외부의 메모리에 접근하기 위하여 포트 0와 포트 2, 두개의 포트가 할당되어 있는데 포트 0는 어드레스 A0~A7과 데이터 D0~D7을 공유하도록 되어 있다.

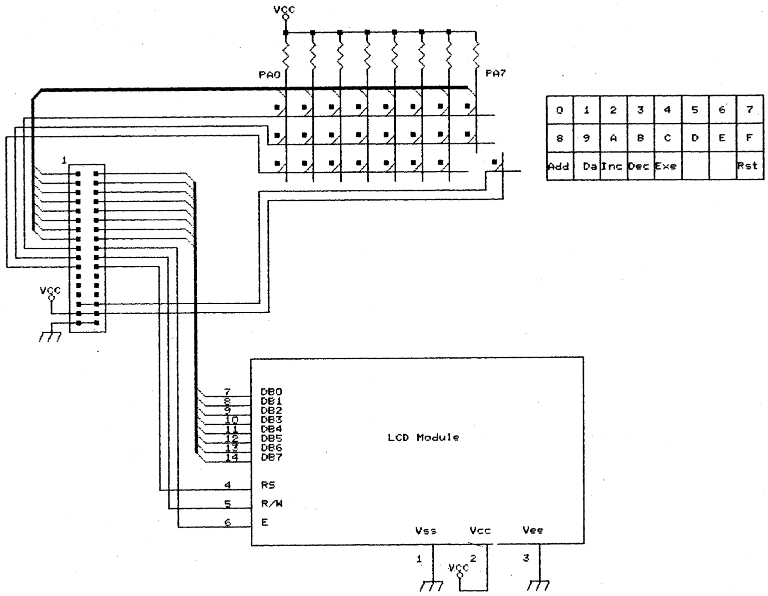
●프로그램메모리

먼저 외부 프로그램 메모리의 리드 사이클에 대하여 알아보자. 이 사이클의 시작은 포트 0에서 어드레스 A0~A7을 만드는 것으로 시작한다. 이때 A8~A15도 메모리 지정에 필요한 어드레스를 만든다. 그 후에 "ALE"(Address Latch Enable) 펄스가 발생되면 A0~A7은 다음의 "ALE" 펄스가 있기 전까지 그림 2에서 LS373의 출력에 고정된다. 그후에 "PSEN"이 Low로 되면 프로그램 메모리의 데이터가 버스를 통해 포트로 입력된다. 이 타이밍 사이클이 이해되면 LS139의 "G"단자가 왜 "PSEN"과 연결되어 있는지와, "ALE"가 왜 LS373의 "G"에 연결되어 있는지를 알 수 있을 것이다.

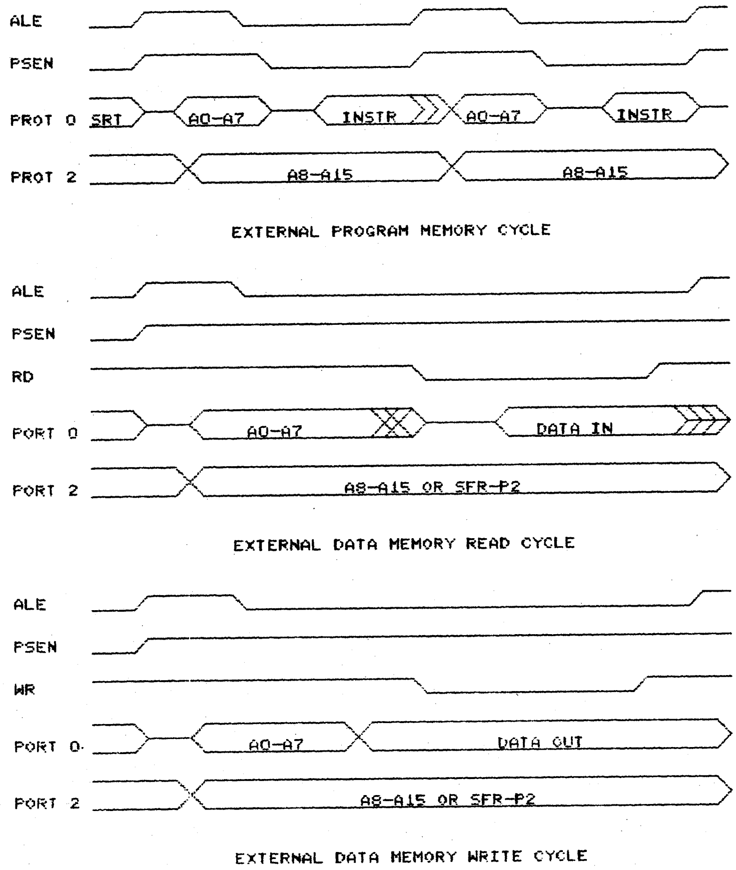
●데이터메모리

이제는 외부 데이터 메모리의 리드, 라이트 사이클에 대하여 알아보자. 그림 4의 가운데와 아래의 것이 각각 리드와 라이트 사이클이다. 이것 역시 어드레스를 만들어내는 과정은 프로그램 메모리의 리드 사이클과 동일하다. 다른 점은 "PSEN"대신에 "RD"와 "WR"이 관계하는 것인데, 이것도 "PSEN"과의 관계를 생각하면 쉽게 알 수 있을 것이다. 이런 관계로 LS139의 또 하나의 "G"에는 "RD"와 "WR"의 AND 출력이 연결되어 있다.

〔그림 3〕 LCD 모듈과 키입력장치

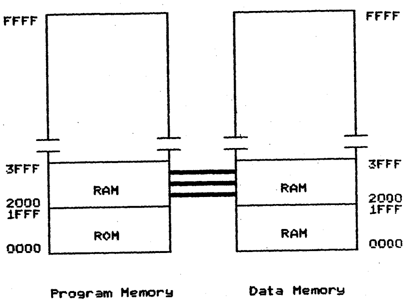


〔그림 4〕 메모리 리드, 라이트 사이클



이는 외부의 데이터 메모리가 호출될 때에만 디코더가 동작되도록 하기 위함이다.

〔그림 5〕 메모리 할당 맵



●메모리구성

롬에는 당연히 시스템의 모니터 프로그램이 들어있고, 두개의 램중에 하나는 데이터 메모리로 사용되고, 나머지 하나는 앞에서 이야기한 프로그램 메모리에 데이터를 쓸 수 없는 것을 해결하기 위하여 적절한 절차로 프로그램 메모리로 혹은 데이터 메모리로 사용할 수 있게 해 놓았다. 이 램의 "RD", "WR", "CS" 단자는 LS157의 출력 "Y1", "Y2", "Y3"에 연결되어 있다.

LS157은 4입력 2채널의 멀티플렉서로 "$\overline{A}$/B"의 입력이 "0"이면 Yn=An이 되고, "$\overline{A}$/B"의 입력이 "1"이 되면 Yn=Bn이 된다. 그림 2에서 볼 수 있듯이 LS157의 A1에는 CPU의 "RD", A2에는 CPU의 "WR", A3에는 데이터 메모리 디코더의 Y1이 연결되어 있어 데이터 메모리 접근에 필요한 제어선이 된다. 그리고 B1에는 "0", B2에는 "1", B3에는 프로그램 메모리 디코더의 Y1이 연결되어 있다. 이것은 프로그램 메모리의 접근에 필요한 조건이다. 또한 LS157의 "$\overline{A}$/B"단자는 LS273의 "Q1"에 연결되어 있다. 이는 뒤에서 설명하게 된다.

그러면 각각의 메모리에 대한 어드

〔표 2〕 LCD 모듈에 대한 명령

명령	코드 RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	설명	실행시간 (최대)
화면소거	0	0	0	0	0	0	0	0	0	1	화면을 소거하고 커서를 어드레스 0에 위치	1.64ms
커서를 홈에 위치	0	0	0	0	0	0	0	0	1	*	커서를 어드레스 0에 위치하고 DDRAM의 내용은 변화없음	1.64ms
모드 고정	0	0	0	0	0	0	0	1	I/D	S	커서 이동방향과 사양을 고정하고 화면을 시프트 안함. 이러한 동작은 데이터를 읽거나 기록하는동안 실행	40μs
화면 ON/OFF 제어	0	0	0	0	0	0	1	D	C	B	화면(D)과 커서(C)를 ON/OFF로 고정. 커서에 있는 문자를 깜빡거림(B)	40μs
커서/화면 시프트	0	0	0	0	0	1	S/C	R/L	*	*	DDRAM의 내용 변화없이 커서이동과 화면시프트	40μs
기능 고정	0	0	0	0	1	DL	N	F	*	*	데이터길이(DL)와 화면라인(L) 그리고 문자폰트(F)를 고정	40μs
CGRAM주소 고정	0	0	0	1	A_{CG}						CGRAM 주소고정. (CGRAM데이터를 주고 받음)	40μs
DDRAM주소 고정	0	0	1	A_{DD}							DDRAM주소고정. (DDRAM데이터를 주고 받음)	40μs
Busy Flag/주소읽음	0	1	BF	AC							내부동작 실행을 표시하는 Busy Flag(BF)와 어드레스 카운터 내용을 읽음	0μs
CGRAM/DDRAM 데이터 기록	1	0	Write Data								DDRAM이나 CGRAM으로 데이터를 기록	40μs
CGRAM/DDRAM 데이터 읽음	1	1	Read Data								DDRAM이나 CGRAM으로부터 데이터를 읽음.	40μs

코드	설명	실행시간(최대)
I/D=1 : 증가 0 : 감소 S=1 : 화면 시프트 포함 S/C=1 : 화면 시프트 0 : 커서 이동 R/L=1 : 오른쪽으로 시프트 0 : 왼쪽으로 시프트 DL=1 : 8bit 0 : 4bit N=1 : 1/16Duty 0 : 1/8, 1/11Duty F=1 : 5×10도트 0 : 5×7도트 BF=1 : 내부동작 실행 0 : 명령 수행	DDRAM : Display Data RAM CGRAM : Character Generator RAM ACG : CGRAM주소 ADD : DDRAM주소에 커서주소를 더함 AC : DDRAM과 CGRAM에서 어드레스 카운터 * : 사용 안함	f_{cp} 또는 f_{osc} =250KHz 주파수변화시 실행시간도 변화 예) f_{cp} 또는 f_{osc} =270KHz $40\mu s \times \frac{250}{270} = 37\mu s$

레스를 알 수 있을 것이다. 우선 프로그램 메모리인 롬은 0000H~1FFFH에 할당되고, 데이터 메모리인 램 또한 0000H~1FFFH에 할당되어 있다. 공용 램의 어드레스는 2000H~3FFFH에 할당되어 있다. 이것을 그림 5에 나타내었다.

I/O

이제 메모리 구조에 대하여 어느 정도 설명이 되었으니 I/O로 이야기를 돌려 보기로 하자. 그림 2에서 볼 수 있듯이 I/O장치는 8255와 LS273 두 종류를 사용하였다. 8255는 인텔의 PPI (Programmable Peripheral Interface)로서 그 사용이 매우 간단한 장치이다. 그리고 LS273은 입력없이 출력만을 필요로 할 때 사용하는 D플립플롭이다.

이들의 어드레스 할당은 앞에서 밝혔듯이 메모리구성이 기능한 I/O이기 때문에 8255는 데이터 메모리의 4000H~4003H에, 그리고 LS273은 6000H에 할당되어 있다.

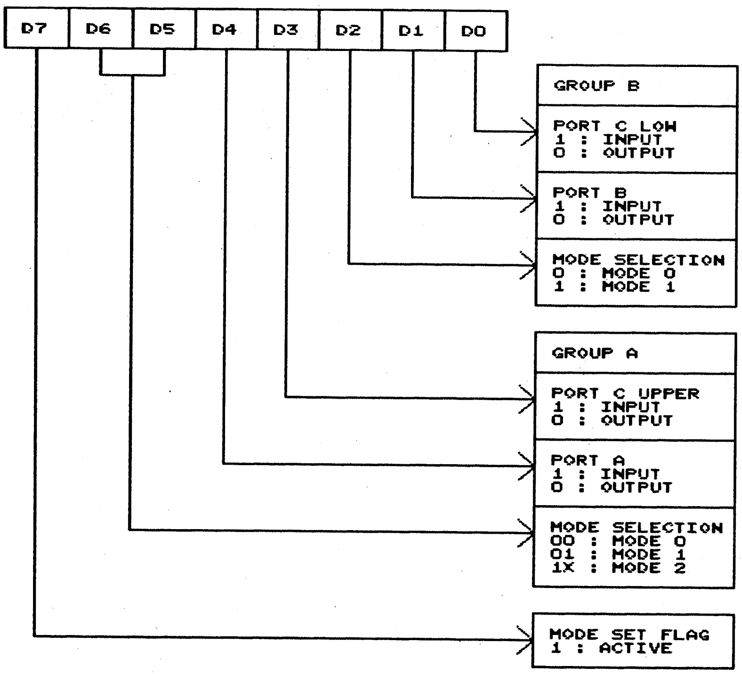
이 장치에 데이터를 보내고 싶을때는 8255를 초기상태로 만들어 주어야 한다. 간단히 설명하면 8255에 할당된 4개의 어드레스중에 마지막 어드레스인 4003H는 제어코드가 들어갈 주소이다. 이 번지에 8255를 어떻게 사용할 것인가하는 정보를 주는 것이 초기화 작업이다. 이 제어코드형식을 그림 6에 나타내었다. 자세한 것은 8255설명서를 참조하기 바란다. 그리고 LS273은 8255보다 간단하여 6000H 데이터를 보내면 출력이 된다. 이를 위한 8031AH의 명령은 다음과 같다.

```
MOV DPTR,#4003H    }
MOV A,(제어워드)    } 8255 초기화
MOVX @DPTR,A       }

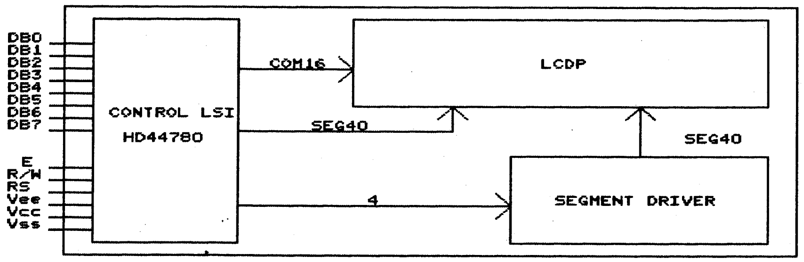
MOV DPTR,#4000H    } 8255 포트A에서 입력
MOVX A,@DPTR       }

MOV DPTR,#400H     } 8255 포트A로 출력
MOVX @DPTR,A       }
```

〔그림 6〕 8255 제어코드 형식



〔그림 7〕 LCD 블록도



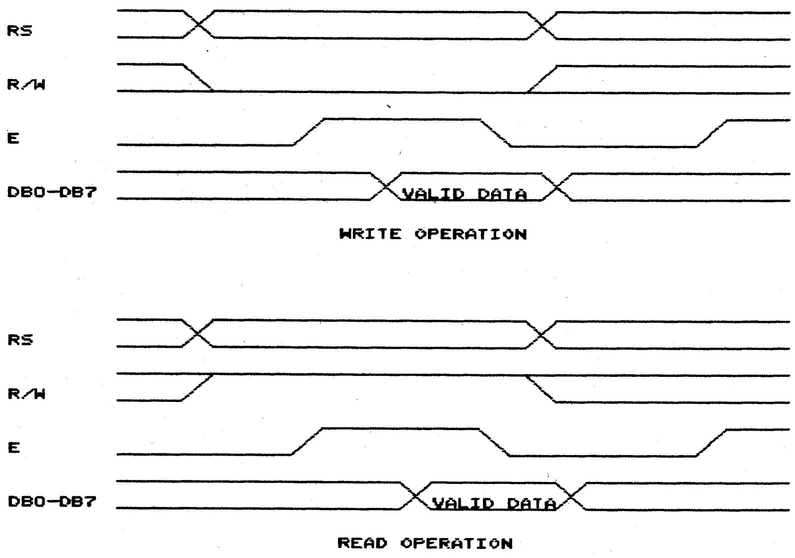

```
MOV DPTR,#6000H    } LS273으로 출력
MOVX @DPTR,A       }
```

다음은 이 시스템에서 사용한 LCD (Liquid Crystal Display) 모듈과 키입력장치에 대하여 알아본다. 먼저 그림 3에서 키입력장치의 구성을 보면 전형적인 매트릭스식이다. 키입력장치에서 회로적으로는 어려운 것이 없고 소프트웨어에서 눌린 키를 찾아낼 때 채터링에 대처해야 한다는 것이다. 그리고 디스플레이 장치인 LCD모듈은 내부에 콘트롤러를 포함하여 외부에서 명령코드와 아스키코드(ASCII Code)만으로 제어를 할 수가 있고, 한번 써넣은 글자는 다음에 써넣을 때까지 그대로 남아있게 된다. 이 LCD모듈에 대한 명령을 표 2에, 블록 다이어그램을 그림 7에, 리드, 라이트 동작의 사이클을 그림 8에, 그리고 핀 배치를 그림 9에 나타내었다. 참조하기 바란다.

메모리 제어

그러면 이 시스템 제작의 제일 중요한 부분이라 볼 수 있는 공용 메모리

〔그림 8〕 LCD 모듈의 리드, 라이트 사이클



〔그림 9〕 LCD 모듈의 핀배치

PIN NO.	SYMBOL	LEVEL	FUNCTION	
1	Vss	–	POWER SUPPLY	0 V (GND)
2	Vcc	–		+5 V
3	Vee	–		for Liquid Crystal Driver
4	RS	H/L	Register Select H : Data Input L : Instruction Input	
5	R/W	H/L	H : Data Read (Module -> CPU) L : Data Write (Module <- CPU)	
6	E	H,H/L	Enable Signal	
7	DB0	H/L	Data Bus Lines	
8	DB1	H/L		
9	DB2	H/L		
10	DB3	H/L		
11	DB4	H/L		
12	DB5	H/L		
13	DB6	H/L		
14	DB7	H/L		

의 선택을 제어하는 방법을 살펴보자. 앞에서 설명한대로 LS157의 An에는 프로그램 메모리의 제어선들이, Bn에는 데이터 메모리의 제어선들이 연결되어 있다. 그러므로 시스템의 모니터 프로그램에서는 프로그램 입력중에서는 Yn을 An과 연결시키고, 실행시킬때는 Yn을 Bn과 연결시켜 준다. 이를 위한 명령은 다음과 같다.

```
MOV DPTR,#6000H  }
MOV A,#00H       }  공용 램을 데이터메모리로 고정
MOVX @DPTR,A     }

MOV DPTR,#6000H  }
MOV A,#01H       }  공용 램을 프로그램메모리로 고정
MOVX @DPTR,A     }
```

이제는 실제 제작에 필요한 몇가지 사항을 밝히고자 한다. (하드웨어적인 경험이 있는 분들은 넘어가도 좋다.) 부품을 기판에 어떻게 배치하느냐에 따라 회로의 배선이 상당히 간편해질 수 있다. 우선 부품을 배치할때 8031AH의 특성상 CPU의 오른쪽에 (핀 번호 21~40) 메모리를 배치하는 것이 배선상 편리하다. 당연한 이야기이지만, 이 쪽에 포트 0와 포트 2가 있기 때문이기도 하다. 그리고 디코더와 멀티플렉서는 CPU의 아래쪽에 배치하고 8255와 LS273은 CPU의 왼쪽에 배치하는 것이 합리적인 것 같다. 그림 10에 부품의 배치도를 나타내었다.

노이즈처리

부품 배치가 끝나면 먼저 전원배선을 하여야 한다. 전원선이 포함된 기판을 사용하는 경우에는 편리해진다. Vcc와 그리고 GND사이에 바이패스 콘덴서라고 하는 노이즈 제거용의 콘덴서를 달아야 한다. 이것은 일반적으로 104(0.1μF)의 세라믹 또는 탄탈 콘덴서를 사용한다. 이 콘덴서의 역할은 직류와 교류, 저주파와 고주파가 동시에 있을 때 교류분만 또는 주파수가 다른 어느 한쪽을 콘덴서에서 제거시킨다. 따라서 노이즈가 자동적으로 제거된다.

컴파일

이쯤하면 시스템에 대한 설명이 대충 끝난 것 같다. 그러면 이제 소프트웨어에 대하여 알아 보기로 하자. 리스트 1의 프로그램을 보면서 생소함을 느끼는 분들이 많았을 것이다.여기서 사용된 언어는 기존의 여러 고급언어와 비슷한 PLM-51인데 처음 접하는 분들이 많으리라 생각한다. 이 PLM-51 컴파일러는 인텔에서 만들어진 것으로, 컴파일과 링크를 하여 2진코드를 16진코드로 변환하는 과정까지 거치면 인텔계 16진 코드형식의 데이터가 만들어진다. 이 데이터를 롬에 프로그램하여 사용해야 한다.

이 PLM-51 컴파일러의 사용법에 대하여 좀더 알아보자. 우선 필요한 화일들을 화면 1에 나타내었다. 이중에서 tb.p51이 소스프로그램이다. 실행시키는 과정은 화면 2에 나타내었다. 일반적으로 우리가 많이 쓰는 에디터인 SK나 PE, NE 등으로 프로그램을 입력한 후 바로 PLM-51로 컴파일시키면 에러가 발생하는데, 이때 이 에러를 없애려면 다음과 같이 실행시키면 된다.

```
>TYPE TB.P51 > TEMP.P51
>DEL TB.P51
>REN TEMP.P51 TB.P51
```

이런 과정을 거쳐야 하는 이유를 자세히는 모르겠으나, 이 컴파일러와 같이 들어있는 에디터를 사용하지 않아서인 것 같다.

이 리스트의 메인 루틴은 뒷부분의 call long_delay;라는 문장부터 실행하는데, 그 아래

```
do while 1;
    .
    .
end;
```

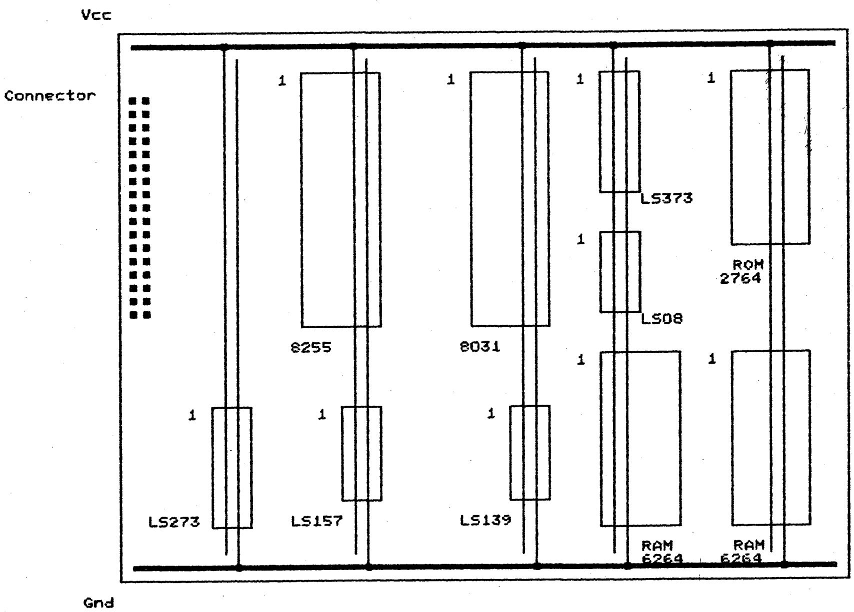
로 묶여져있는 루틴이 2개 들어 있다. 이 루틴중에

```
    .
    .
if status_in = 20
   then do;
        .
        .
        system_status = 01h;
        .
        .
        system_status = 00h;
        .
        .
        end;
    .
    .
```

이 두개의 명령이 공용램을 프로그램 메모리와 데이터 메모리의 변화에 사용하는 것이다. 위의 명령이 공용 램을 프로그램 메모리로, 아래의 명령이 데이터 메모리로 고정시키는 것이다.

프로그램의 앞부분에 call time( ); 이라는 프로시듀어는 PLM-51 컴파일러에서 지원하는 표준 프로시듀어이다. 여기에서 이런 시간지연이 필요한 이유는 LCD모듈이 CPU의 속도를 따라

〔그림 10〕 부품배치도



〔화면 1〕 PLM-51 컴파일에 사용된 화일

```
A>dir

 Volume in drive A is PERSONALCOMPUTER
 Directory of  A:\

COMMAND  COM    23612   5-31-1989  12:00a
RL51     EXE    51774   5-31-1989  12:00a
OH       EXE    19293   5-31-1989  12:00a
PLM51    EXE   119965   5-31-1989  12:00a
PLM51    LIB    11571   5-31-1989  12:00a
REG51    DCL     2243   5-31-1989  12:00a
TB       LST    13638   5-31-1989  12:00a
TB       P51     8717   5-31-1989  12:00a
TB       M51     1797   5-31-1989  12:00a
TB       OBJ     3250   5-31-1989  12:00a
TB       ABS     1358   5-31-1989  12:00a
TB       HEX     3533   5-31-1989  12:00a
        12 File(s)     43008 bytes free

A>
```

〔화면 2〕 PLM-51 컴파일 실행과정

```
A>plm51 tb.p51


DOS 3.20 (038-N) PL/M-51 COMPLIER V1.2
Copyright 1982,1983,1986 Intel Coporation
PL/M-51 COMPILATION COMPLETE.      0 PROGRAM ERROR(S)


A>R151 TB.OBJ,PLM51.LIB TO TB.ABS


DOS 3.20 (038-N) MCS-51 RELOCATOR AND LINKER V3.0
Copyright 1982,1983,1986 Intel Corporation

A>OH TB.ABS TO TB.HEX

DOS 3.20 (038-N)  OH  V1.0
Copyright 1986 Intel Coporation

A>
```

갈만큼 충분히 빠르지 못하기 때문이다.

사용법

제작이 끝나면 전원을 넣는다. 제대로 만들어져 동작한다면, LCD모듈의 윗줄에는 "Modify Memory"라는 글자가 나오고, 아래줄에는 "Add:2000 Data:XX"라고 나타난다. XX는 램에 어떤 값이 있을지 모르기 때문에 XX라고 한 것이다. 그림 3의 키입력장치 배치에서 위의 두줄은 0~F까지의 16진수 숫자를 의미하고, Add는 어드레스를, Da는 데이터를, RST는 시스템을 리세트하는 키이며, Exe는 현재 가리키는 어드레스부터 프로그램을 실행시키는 키이다.

Add키를 누른후 숫자키를 치면 어드레스를 새로이 지정할 수 있다. 이후에 Da키를 누르면 다음부터 입력되는 숫자는 데이터값을 변화시키게 된다. Inc와 Dec키는 각각 어드레스를 증가 또는 감소시킨다. 이렇게 프로그램을 다 입력시킨 후 Exe키를 누르면 화면 윗줄에 "Execute Program"이라는 글자가 나오고 프로그램을 실행한다. 만약 프로그램의 실행시간이 짧으면 너무 빨리 없어지므로 보이지 않을 수도 있다.

한가지 주의할 점은 프로그램이 실행되고 다시 모니터 프로그램으로 돌아오기위해 프로그램의 끝에 "RET" 명령을 넣어 주어야 한다는 것이다.

맺음말

이 시스템을 만들고 나서 몇가지 미련이 남는 점이 있는데, 우선 직렬통신을 위한 RS-232C나 RS-422A 인터페이스 칩을 넣지 않은 것이고, 레지스터의 값을 알아 볼 수 있는 기능을 넣지 않은 것, 인터럽트에 대비하여 롬안에 존재하는 벡터어드레스에서 공용 램의 특정어드레스를 호출할 수 있도록 처리하지 않은 것 등이다. 이점은 관심있는 독자들께서 해결하여 주셨으면 한다. PC

〈참고문헌〉

1. Micro controller handbook-Intel
2. Component data catalog-Intel
3. PLM-51 user's guide-Intel
4. MCS-51 utility user's guide for DOS-Intel
5. The bipolar digital integrated circuit databook for design engineers-Texas Ins.

〔리스트 1〕

```
$title ('MCS-51 start kit management system')

main_prog:    do;

declare scan_sequence(8) byte constant(0feh,0fdh,0fbh,
                         0f7h, 0efh,0dfh,0bfh,07fh);
declare PPI_A byte at(4000h) auxiliary;
declare PPI_B byte at(4001h) auxiliary;
declare PPI_C byte at(4002h) auxiliary;
declare PPI_CW byte at(4003h) auxiliary;

declare system_status byte at(6000h) auxiliary;

declare key_flag byte;

declare mem_address word;
```

```
declare status_in byte;

$nolist
$include (reg51.dcl)
$list

system_init:  procedure;
              mem_address = 2000h;
              system_status = 00h;
              PPI_CW = 90h;   /* ai bo co */
              PPI_C = 00000000b;
              PPI_C = 00010000b;
              call time(1);
              PPI_B = 00001100b;
              PPI_C = 00000000b;
              call time(50);
              PPI_C = 00000000b;
              PPI_C = 00010000b;
              call time(1);
              PPI_B = 00111000b;
              PPI_C = 00000000b;
              call time(10);
end           system_init;

char_out:     procedure(ascii);
declare ascii byte;
declare busy_flag byte;
              PPI_CW = 92h;   /* ai bi co */
              do while 1;
                   PPI_C = 00100000b;
                   PPI_C = 00110000b;
                   call time(1);
                   busy_flag = PPI_B;
                   PPI_C = 00000000b;
                   if busy_flag < 80h then goto out_
                                        char;
              end;
out_char:
              PPI_CW = 90h;   /*ai bo co */
              PPI_C = 01000000b;
              PPI_C = 01010000b;


$title ('MCS-51 start kit management system')

main_prog:   do;

declare scan_sequence(8) byte constant(0feh,0fdh,0fbh,0f7h,
                                 0efh,0dfh,0bfh,07fh);
declare PPI_A byte at(4000h) auxiliary;
declare PPI_B byte at(4001h) auxiliary;
declare PPI_C byte at(4002h) auxiliary;
declare PPI_CW byte at(4003h) auxiliary;

declare system_status byte at(6000h) auxiliary;

declare key_flag byte;

declare mem_address word;

declare status_in byte;

$nolist
$include (reg51.dcl)
$list

system_init:  procedure;
              mem_address = 2000h;
              system_status = 00h;
              PPI_CW = 90h;   /* ai bo co */
              PPI_C = 00000000b;
              PPI_C = 00010000b;
              call time(1);
              PPI_B = 00001100b;
              PPI_C = 00000000b;
              call time(50);
              PPI_C = 00000000b;
              PPI_C = 00010000b;
              call time(1);
              PPI_B = 00111000b;
              PPI_C = 00000000b;
              call time(10);
end           system_init;

char_out:     procedure(ascii);
declare ascii byte;
declare busy_flag byte;
              PPI_CW = 92h;   /* ai bi co */
              do while 1;
                   PPI_C = 00100000b;
                   PPI_C = 00110000b;
                   call time(1);
                   busy_flag = PPI_B;
                   PPI_C = 00000000b;
                   if busy_flag < 80h then goto out_char;
              end;
out_char:
              PPI_CW = 90h;   /*ai bo co */
              PPI_C = 01000000b;
              PPI_C = 01010000b;
              call time (1);
              PPI_B = ascii;
              PPI_C = 00000000b;
end           char_out;
```

```
locate:       procedure(x_position,y_position);
declare x_position byte;
declare y_position byte;
declare out_value byte;
declare busy_flag byte;
              if x_position = 0 then out_value = 10000000b;
                    else out_value = 11000000b;
              out_value = out_value + y_position;
              PPI_CW = 92h;   /* ai bi co */
              do while 1;
                  PPI_C = 00100000b;
                  PPI_C = 00110000b;
                  call time(1);
                  busy_flag = PPI_B;
                  PPI_C = 00000000b;
                  if busy_flag < 80h then goto out_addr;
              end;
out_addr:
              PPI_CW = 90h;   /*ai bo co */
              PPI_C = 00000000b;
              PPI_C = 00010000b;
              call time(1);
              PPI_B = out_value;
              PPI_C = 00000000b;
end           locate;

press_check:  procedure byte;
declare out_count byte;
declare in_count byte;
          do out_count = 0 to 2;
              PPI_C = scan_sequence(out_count) and 0fh;
              do in_count = 0 to 7;
                  if PPI_A = scan_sequence(in_count)
                   then return out_count*8 + in_count;
              end;
          end;
          return 0ffh;
end       press_check;

keyin_proc:   procedure byte;
declare key_count byte;
declare temp_key byte;
              key_count = 0;
              do while 1;
                  temp_key = press_check;
                  if temp_key = 0ffh then
                  do;
                      key_flag = 0;
                      key_count = 0;
                  end;
                  if temp_key <> 0ffh then
                  do;
                      key_count = key_count + 1;
                      if key_count > 10 and key_flag = 0
                          then  do;
                                  key_flag = 1;
                                  return temp_key;
                               end;
                  end;
              end;
end           keyin_proc;

string_out:   procedure(string_ptr_1,string_ptr_2);
declare (string_ptr_1,string_ptr_2) word;
declare schar1 based string_ptr_1 byte constant;
declare schar2 based string_ptr_2 byte constant;
declare string_off byte;
              do string_off = 0 to 15;
                  call locate(0,string_off);
                  call char_out(schar1);
                  string_ptr_1 = string_ptr_1 + 1;
              end;
              do string_off = 0 to 15;
                  call locate(1,string_off);
                  call char_out(schar2);
                  string_ptr_2 = string_ptr_2 + 1;
              end;
end           string_out;

bin_to_ascii: procedure(binary) byte;
declare binary byte;
              if binary <= 9 then return binary + 30h;
                     else return binary + 37h;
end           bin_to_ascii;

dsp_ad_da:    procedure;
declare dsp_ascii byte;
declare mem_data based mem_address byte auxiliary;
              dsp_ascii = low(mem_address);
              dsp_ascii = dsp_ascii and 0fh;
              call locate(1,7);
              call char_out(bin_to_ascii(dsp_ascii));
              dsp_ascii = low(mem_address);
              dsp_ascii = shr(dsp_ascii,4);
              call locate(1,6);
              call char_out(bin_to_ascii(dsp_ascii));

              dsp_ascii = high(mem_address);
              dsp_ascii = dsp_ascii and 0fh;
              call locate(1,5);
              call char_out(bin_to_ascii(dsp_ascii));
              dsp_ascii = high(mem_address);
              dsp_ascii = shr(dsp_ascii,4);
```

```
            call locate(1,4);
            call char_out(bin_to_ascii(dsp_ascii));

            dsp_ascii = mem_data;
            dsp_ascii = dsp_ascii and 0fh;
            call locate(1,15);
            call char_out(bin_to_ascii(dsp_ascii));
            dsp_ascii = mem_data;
            dsp_ascii = shr(dsp_ascii,4);
            call locate(1,14);
            call char_out(bin_to_ascii(dsp_ascii));

end         dsp_ad_da;

addr_modify:  procedure(low_addr);
declare low_addr byte;
declare match_addr word;
            match_addr = double(low_addr);
            mem_address = shl(mem_address,4);
            mem_address = mem_address or match_addr;
end         addr_modify;

data_modify:  procedure(low_data);
declare low_data byte;
declare mem_data based mem_address byte auxiliary;
            mem_data = shl(mem_data,4) or low_data;
end         data_modify;

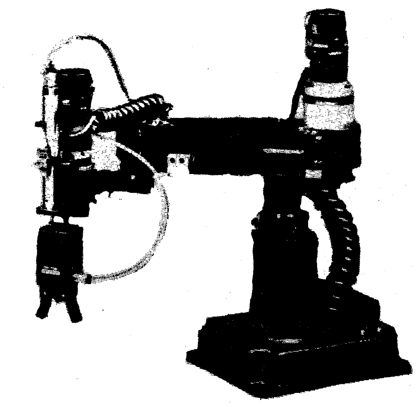
long_delay:   procedure;
declare delay_count byte;
            do delay_count = 0 to 20;
                call time(200);
            end;
end         long_delay;

            call long_delay;
            call system_init;
            call string_out(.(' Modify Memory. '),
                            .('Add:xxxx Data:xx'));

data_mode:
    do while 1;
        call dsp_ad_da;
        status_in = keyin_proc;
        if status_in = 16 then goto addr_mode;
        if status_in = 17 then ;
        if status_in = 18 then mem_address = mem_address + 1;
        if status_in = 19 then mem_address = mem_address - 1;
        if status_in = 20
          then do;
                call string_out(.('Execute program.'),
                                .('                '));
                system_status = 01h;
                call mem_address;
                system_status = 00h;
                call string_out(.(' Modify Memory. '),
                                .('Add:xxxx Data:xx'));
              end;
        if status_in <= 15 then call data_modify(status_in);
    end;
addr_mode:
    do while 1;
        call dsp_ad_da;
        status_in = keyin_proc;
        if status_in = 16 then ;
        if status_in = 17 then goto data_mode;
        if status_in = 18 then mem_address = mem_address + 1;
        if status_in = 19 then mem_address = mem_address - 1;
        if status_in = 20
          then do;
                call string_out(.('Execute program.'),
                                .('                '));
                system_status = 01h;
                call mem_address;
                system_status = 00h;
                call string_out(.(' Modify Memory. '),
                                .('Add:xxxx Data:xx'));
              end;
        if status_in <= 15 then call addr_modify(status_in);
    end;

end      main_prog;
```

마이크로프로세서를 이용한
FA SYSTEM 과정
(생산자동화)



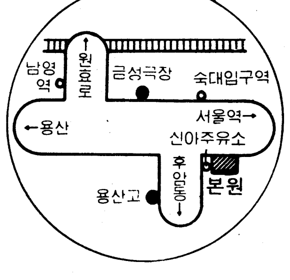
※ 개강 : 매월 5일

講義案內 (2개월 완성)

본원에서는 다년간 Computer Hardware 과정을 교육하여 많은 수료생을 배출, 각계에서 공헌하고 있는 바 금번 생산업계의 요청에 따라 컴퓨터 프로세서를 이용한 생산체제와 산업용 로보트의 발전으로 제품과 제조공정의 설계 및 계획, 공장 현장에 있는 생산 제조장치의 컴퓨터 제어, 원료 및 자재의 유동제어, 공간 활용 및 작업계획 등의 조직적 기획 등의 필요성에 따른 FA 과정의 이론, 실기를 위주로 특강을 하게 되었으니 관계 직원의 많은 참여 바랍니다.

■ **강의일정**

주별	일자 \ 과정	강습내용 (이론 및 실기)	주별	일자 \ 과정	강습내용 (이론 및 실기)
1주	1일차	OP Amp 응용회로	5주	1일차	DC 모우터의 특성
	2일차	필터 설계(1)-RLC 필터		2일차	DC 모우터의 구동법-PWM 제어
	3일차	필터 설계(2)-액티브 필터		3일차	Micom을 이용한 DC 모우터 제어
2주	1일차	TTL회로 설계	6주	1일차	Step 모우터의 특성
	2일차	A/D변환		2일차	Step 모우터 제어회로
	3일차	D/A변환		3일차	Micom을 이용한 Step 모우터 제어
3주	1일차	센서 활용 I-광센서	7주	1일차	IBM PC H/W
	2일차	센서 활용 II-초음파, 자기센서		2일차	IBM PC 병렬 인터페이스
	3일차	센서 인터페이싱		3일차	PC통신-통신방식 및 통신규약
4주	1일차	Remote Control 시스템 설계	8주	1일차	RS232C Serial 인터페이스
	2일차	Remocon 송신부 설계(PPM 변조)		2일차	스위칭 전원의 설계(I)
	3일차	Remocon 수신부 설계 (PPM 복조 및 코드 해석)		3일차	스위칭 전원의 설계(II)



전자산업 4반세기를 지켜온 기술인의 전당

과인 **한국전자통신학원** 컴퓨터 기술부

(구 한국전파학원)

140 서울시 용산구 갈월동 17번지 ☎ 754-1751/2902/1044

•유틸리티 소개•

강력한 도스쉘 기능의 1DIR

정승민 / 아주대 수학과 HAL

1DIR은 화일의 관리와 선택, EXE, CON, BAT 화일의 실행 및 배치화일의 작성, 시스템 보완을 위한 도스쉘 기능을 누구나 쉽게 사용할 수 있도록 한 유틸리티로서 노턴코맨더, PCTOOLS, 퀵도스, XTREE등과 함께 많이 사용되는 프로그램중의 하나이다.

PC하면 IBM-PC가 생각나는 작금의 현실은 여러가지 이유 때문이기도 하지만 바로 IBM-PC의 개방구조와 풍부한 소프트웨어에 있다고 본다(실상 애플의 신화도 이런 이유에서 나왔다고 본다). 특히 현재 IBM-PC에서 작동하는 소프트웨어는 셀 수가 없을 정도이다.

예를 들어서 스프레드 시트만 하더라도 과거 애플 시절에는 비지캘크 정도가 고작이던 것이 IBM-PC 시절(?)에는 로터스를 위시하여 쿼트로 등등 선택의 폭이 상당히 넓다. 이러한 현상은 업무용 패키지에만 국한되지 않고 컴퓨터 랭귀지 컴파일러(특히 마이크로 소프트사와 볼랜드사), 유틸리티, 통신 소프트웨어 등 많은 부분에서 벌어지고 있다.

유틸리티하면 노턴이 생각나던 시절도 있었지만 최근에는 각 소프트웨어 하우스가 세련되고 편리한 프로그램을 많이 만들어서 팔고 있기 때문에 노턴씨도 별로 빛을 발하지는 못하고 있는 실정이다.

이번 호에 소개할 1DIR은 소위 도스 쉘이라는 유틸리티의 일종으로서 다른 회사의 것으로 유명한 것들은 노턴사의 노턴 코맨더, 센트럴사의 PC-TOOLS, 가즐사의 퀵도스, 이그제큐티브사의 XTREE 등이 있다. 위의 것들은 모두 다 각각 나름대로의 특징이 있는 데 이번에는 부바키사의 1DIR을 소개하겠다.

특징

도스 쉘 프로그램에는 여러가지가 있으나 1DIR은 도스 쉘 프로그램중에서도 단연 뛰어난 기능을 자랑한다. 1DIR로 무슨 도스 쉘 기능이 있냐고 묻는 대신에 도스기능 중에 할 수 없는 것이 무엇이냐고 묻는 게 더 나을 정도이다. 즉, 1DIR은 초보자, 중급자, 숙련자 모두에게 뛰어난 기능을 제공한다. 그 기능의 예를 든다면 각 레벨별로의 메뉴를 각자가 작성하여 사용할 수 있고 도스 쉘로서의 거의 모든 기능을 제공한다.

도스 쉘로서의 대표적인 예를 든다면 우선 화일 관리, 화일의 선택(태그라고도 한다), EXE, COM, BAT 등의 화일의 실행, 배치화일의 작성(특히 1DIR은 배치화일 작성 기능이 뛰어나다), 각 메뉴에 패스워드를 걸어서 시스템에 보안을 줄 수 있고, 몇 분 동안 유저가 아무 것도 하지 않으면 모니터의 보호를 위해서 스크린을 끄는 등의 기능이 있는 데, 물론 1DIR은 이 모든 것을 지원한다.

앞서 설명한 것처럼 어떻게 보면 복잡하기까지 한 기능들의 나열을 보고 독자들은 1DIR이 도스의 고도 숙련자만이 사용하는 것인줄 오해를 할 것이다. 하지만 1DIR은 유저가 초보자이든, 숙련자이든 편리한 도스 쉘 기능을 가지고 있다.

저급 워드프로세서계의 잘난체 하는(?) 워드퍼펙과는 달리, 1DIR의 정책은 모든 도스의 사용자에게 촛점을 맞추어서 제작된 것이다. 하지만 도스 쉘 프로그램은 앞에서도 나왔지만 1DIR만이 있는 것이 아니다. 1DIR을 구하기 전에 필자가 애용했었던(그리고 지금도 1DIR과 같이 애용하는) 노턴 코맨더나 PCTOOLS(이 프로그램은 많은 초보자들에게 지금도 널리 사용되어지고 있다), 또 퀵도스(이 역시 많은 사용자를 확보하고 있다), XTree(이 프로그램은 조금 숙련된 유저들이 애용하는 경향이 있다) 등도 나름대로의 사용자층을 형성하고 있다.

하지만 다른 도스 쉘용 프로그램들이 초보자나 숙련자만을 위한 특정 계층에 촛점이 맞추어진 데 비하여 1DIR은 거의 모든 도스 사용자층을 커버하는 것이 큰 특징이다.

미국에서의 판매가격도 1DIR이 도스 쉘중에서 제일 높다. 가격이 높다는 소리는 보통 그만큼 기능이 강하다는 이야기와 상통한다.

필자가 1DIR을 사용하면서 가장 편리하다고 느낀 것은 배치 화일을 만들어 주는 "배치 빌더"이다. 배치 빌더는 1DIR을 사용하는 모든 과정을 배치화일로 만들어 주어서, 그 다음부터 생성된 배치화일만 실행시키면 작업이 자동

적으로 처리되는 것이다.

인스톨

1DIR의 거의 모든 기능에 만족하는 필자이지만 1DIR에 한가지 불만이 있다면 인스톨이 약간 불편하다는 점이다. 물론 인스톨용 배치화일이 있지만 최근에 개발된 볼랜드사 랭귀지 컴파일러의 세련된 인스톨이나 마이크로소프트 윈도우의 인스톨에 맛들은(?) 유저들에게는 약간 불편을 느끼게 만든다. 하여튼 1DIR 디스켓을 드라이브 A:에 넣고 다음과 같이 하면 된다.

1DIRINST 대상드라이브

예를 들어서 드라이브 B:에 인스톨하고 싶으면

1DIRINST B

와 같이 하면 된다.

여기서 주의할 점은 대상드라이브명 뒤에 콜론(:)을 쓰지 않는다는 점이다. 또 1DIR을 실행시키는 데는 1DIR.COM만 있어도 된다. 1DIR 디스켓에서 1DIR.COM 화일만을 따로 복사해서 자신이 자주 사용하는 플로피 디스크나 하드 디스크의 서브디렉토리에 넣어서 하드 디스크의 경우 PATH를 걸어 사용해도 지장이 없다. 단, 1DIR의 옵션을 자주 바꾸는 유저는 옵션을 바꾸어 주는 프로그램인 OPTIONS.EXE도 같은 디렉토리나 드라이브에 넣어두면 편리하다.

인스톨이 끝나면 AUTOEXEC.BAT에 자동적으로 1DIR이 삽입된다. 그 다음부터는 부팅만 하면 바로 1DIR이 실행되는 것이다.

프로그램의 실행

이제 1DIR이 인스톨된 디스크로 부팅을 시켜보자(아니면 1DIR을 바로 실행시켜도 된다). 1DIR을 실행시키게 되면 화면 1과 같이 메뉴가 나타나고

〔화면 1〕 1DIR 메뉴화면

```
Drive C  | Name    | Ext | Size      | Statistics                 | Toggles
Select ▶ | #### PC # | ### | VOLUME  | ▶ Disk Usage ◀             | Main Menu
         | #1      |     | ▶SUB-DIR  |   3 Hidden files           | Caps  | Print
         | #2      |     | ▶SUB-DIR  |   9 User files             | Batch | Edit
         | #3      |     | ▶SUB-DIR  | 409600 bytes left          | Set-up
         | ARCHIV  |     | ▶SUB-DIR  | 178176 bytes used          | Pause | On
         | BASIC   |     | ▶SUB-DIR  | 21309440 bytes total       | Sort  | Name
         | CAD     |     | ▶SUB-DIR  | ▶▶ Memory Usage ◀◀         | Default | C:
         | COMM    |     | ▶SUB-DIR  | 500848 bytes left          | Display | C:
         | COP     |     | ▶SUB-DIR  | 154512 bytes used          |
         | DOS     |     | ▶SUB-DIR  | 655360 bytes total         |
         | DRAW    |     | ▶SUB-DIR  | ▶▶▶ Today Is ◀◀◀           |
         | DRHALO  |     | ▶SUB-DIR  | Tuesday the 1st            |
         | GRAPHIC |     | ▶SUB-DIR  | 12:04:48am                 |
         | IMSI    |     | ▶SUB-DIR  |                            |
         | JSM     |     | ▶SUB-DIR  |                            |
C>
Compose | Copy | Type | Rename | Erase | Date | Mkdir | Options
The 1 DIR Version 3.50E - Copyright (c) Bourbaki, Inc. 1984, 1985
```

1DIR의 큰 특징중에 하나인 자세한 시스템 상태가 표시된다.

메뉴의 오른쪽 위를 보면 토글상태가 표시되어 있다. 우선 대/소문자 상태인 캡스록이 표시되어 있고, 캡스록 바로 옆에 〈Print〉라고 써있다. 〈Print〉는 프린터 에코우의 온/오프를 결정하는 것이다. 이것이 온되어 있으면 1DIR의 타이프 명령으로 타이프되는 화일의 내용이 프린팅되어 나오게 된다. 이것을 토글시키려면 〈Ctrl-PrtSc〉키를 누르면 된다.

〈Print〉 바로 위는 메인 메뉴 표시이다. 1DIR에서는 메뉴를 각 유저가 자기에게 맞게 수정할 수 있기 때문에 현재 유저가 어느 메뉴를 사용하고 있는지 표시해 준다.

캡스록 표시 바로 아래는 배치 빌더의 온/오프 여부를 표시해 주는 것이다. 배치 빌더는 1DIR에서 하는 명령 하나하나를 배치화일로 기록하게 된다.

배치 빌더 옆에는 '에디트'표시가 있다. 이것이 온되어 있으면 펑션키가 도스의 에디트키로 작용하고 오프되어 있으면 각 펑션키는 해당 1DIR의 명령을 수행하게 된다.

메뉴의 오른쪽 아래를 보면 셋업〈set-up〉 표시가 있는데 셋업 항목으로 커서를 이동시키기 위해서는 〈Ctrl-Q〉나 〈Scroll Lock〉키를 누르면 된다. 그리고 아래위 화살표로 이동해서 해당 항목을 고치면 된다.

〈Pause〉라는 것은 각 1DIR의 명령을 수행한 후에 한번의 키입력을 기다리게 하는 옵션이다. 이것이 오프되어 있으면 해당 1DIR의 명령을 수행하고 바로 메뉴로 되돌아오게 되는 것이다. 따라서 화일의 복사나 삭제후에는 바로 메뉴로 되돌아 오는 게 좋지만 화일을 타이핑할 때는 Pause가 온되어 있는 게 낫다.

Pause 밑에는 Sort가 있다. Sort난은 화일의 디렉토리를 어떤 식으로 소트하여 표시할 것인가에 대하여 정해주는 것이다. 소팅 방법으로는 화일이름, 화일의 확장명, 화일의 날짜, 화일의 크기로 하는 방법이 있다. 소팅 방법의 지정은 〈N〉이나 〈+〉를 누르면 화일이름으로 소팅하는 것이 되고, 〈E〉나 〈−〉키를 누르면 화일의 확장명으로 소

팅하게 되고, 〈D〉나 〈Ins〉키를 누르면 화일의 날짜로 소팅하게 되고, 〈S〉나 〈Del〉키를 누르면 화일의 크기로 소팅하게 된다.

Sort의 밑에는 Default가 있는데, 이것은 현재 표시되는 디렉토리나 드라이브를 말한다. 이것을 바꾸면 자동적으로 바로 밑의 디스플레이도 따라서 바뀐다.

Default 밑의 Display는 1DIR의 디스플레이를 조정할 수 있다. 우선 현재의 리스트가 나온 드라이브를 바꿀 수 있다. 또, 화일의 와일드카드를 사용하여 일부분만 표시할 수도 있다. 디스플레이에 커서가 있을 때 〈*〉키를 누르게 되면 와일드카드 문자를 입력받는다. 이때 유저가 원하는 와일드 문자를 입력하면 해당 화일만이 표시된다.

화면 중앙의 'Statistics'는 〈+〉키와 〈—〉키를 사용하여 디스크의 상태와 메모리 상태의 표시대신에 화일의 날짜를 표시하게 할 수 있다. 즉, 〈+〉키를 누르면 시스템 상태가 표시되고 〈—〉키를 누르면 **화면 2**와 같이 화일의 날짜가 표시된다. 이때 표시되는 날짜는 좌측에 표시된 화일의 행과 일치하는 화일의 날짜가 표시된다.

메뉴의 화면 아래를 보면 8개의 박스가 있고 'Compose'부분이 반전되어 있을 것이다. 이 반전된 것을 '코맨드 커서'라 하고 좌측 화일 표시부분에 반전된 부분을 '화일커서'라 한다. 코맨드커서는 좌우 화살표키를 이용하여 움직이고 화일커서는 아래위 화살표키를 이용하여 움직이게 된다.

1DIR의 메뉴 하단에 나타난 8개의 코맨드를 실행시키는 방법은 좌우 화살표키를 이용하여 원하는 코맨드로 코맨드커서를 이동한 후에 엔터키를 누르면 된다. 혹은 8개의 박스는 좌측에서부터 펑션키가 〈F1〉에서 〈F8〉까지 할당되어 있으므로 각 해당 펑션키를 누르면 된다. 그런데 어떤 코맨드는 코맨드의 수행에 화일이름이 필요한 경우가 있고 어떤 코맨드는 아무 것도 필요없

〔화면 2〕 Statistic에서 〈—〉키를 눌렀을 때

Drive C	Name	Ext	Size	Date	Time
	#### PC #	###	VOLUME	Jan 01, 1980	00:45am
	#1		▶SUB-DIR	Jan 01, 1980	02:43am
	#2		▶SUB-DIR	Jan 01, 1980	02:43am
	#3		▶SUB-DIR	Jan 01, 1980	03:16am
	ARCHIV		▶SUB-DIR	Jan 01, 1980	00:21am
	BASIC		▶SUB-DIR	Aug 02, 1988	04:34pm
	CAD		▶SUB-DIR	Aug 03, 1988	11:08am
	COMM		▶SUB-DIR	Jan 01, 1980	00:02am
	COP		▶SUB-DIR	May 18, 1989	03:46am
	DOS		▶SUB-DIR	Jan 01, 1980	00:00am
	DRAW		▶SUB-DIR	Jan 01, 1980	00:01am
	DRHALO		▶SUB-DIR	Jan 01, 1980	00:00am
	GRAPHIC		▶SUB-DIR	Aug 02, 1988	04:05pm
	IMSI		▶SUB-DIR	Jan 01, 1980	00:14am
	JSM		▶SUB-DIR	Jan 01, 1980	00:00am

Toggles	
Main Menu	
Caps	Print
Batch	Edit
Set-up	
Pause	On
Sort	Name
Default	C:
Display	C:

C>

Compose	Copy	Type	Rename	Erase	Date	Mkdir	Options

The 1 DIR Version 3.50E - Copyright (c) Bourbaki, Inc. 1984, 1985

이 바로 실행되는 것이 있다.

화일이름이 필요한 코맨드에는 'Compose', 'Copy' 등이 있는 데 이때는 화일커서 왼쪽에 'select'라는 표시가 있다. 하지만 'Date', 'Mkdir' 등과 같이 코맨드의 수행에 화일이름이 필요하지 않은 것은 'select'표시가 화일커서 왼쪽에 표시가 되지 않는다.

코맨드

앞에서 1DIR에는 기본적으로 8개의 코맨드가 있다는 것을 설명하였다. 그러면 이들 코맨드들의 기능에 관하여 알아보자.

(1) Compose

Compose는 8개의 코맨드중에서 가장 유용한 것으로 이 모드는 도스 프롬프트 상태와 아주 유사하다. 이 모드에서는 바로 도스의 명령을 입력하고 엔터키를 누르면 그 도스 명령이 실행된다. 또 하나 편리한 것은 화일 커서가 가르키는 화일의 이름과 확장명을 도스 코맨드 라인으로 가져올 수 있다는 것이다. 즉, 뉴메릭 키보드에 있는 〈+〉키를 누르면 화일커서가 가르키는 화일의 이름이 코맨드 라인으로 내려오고, 〈—〉키를 누르면 화일커서가 가르키는 화일의 확장명이 코맨드 라인으로 내려오게 된다.

그리고 그냥 엔터키만을 누르면 화일커서가 가르키고 있는 화일이 실행된다. 또 한라인에 여러 명령을 동시에 줄 수 있는 데, 이것은 각 코맨드를 역인용부호(ʻ)로 구분해주면 된다.

예를 들어 도스에서 'CHKDSK/F'라는 명령을 주는 경우를 생각해 보자. 우선 화일 커서를 움직여서 CHKDSK.COM을 가르키게 한다. 화일 커서의 이동에서 서브디렉토리로 들어갈 때에는 화일커서를 서브디렉토리로 가게 하고 엔터키를 누르면 된다.

화일커서가 CHKDSK.COM을 가르키게 했으면 뉴메릭 키보드의 〈+〉키를 눌러서 코맨드 라인에 'CHKDSK'가 나오게 한다.

여기에서 직접 '/F'를 입력하면 코맨드 라인에는 'CHKDSK/F'라고 표시된다.

이때 엔터키를 누르면 바로 명령이 실행된다.

(2) Copy

화일이나 디렉토리를 복사할 때에 사용한다. 커서를 Copy에 옮긴 후 엔터키를 누르면 대상드라이브명을 묻는다. 이때 대상드라이브명을 입력하고 엔터키를 누르면 복사가 시작된다.

뒤의 Type명령이나 화일삭제명령인 Erase에서도 마찬가지이지만 복사명령은 여러개의 화일을 동시에 처리해야 할 경우가 많다. 이럴 경우 여러개의 화일을 지정해야 하는데 이때 사용하는 키가 바로 〈+〉키와 〈-〉키이다. 〈+〉키는 현재의 화일커서에 있는 화일을 지정할 때 사용하고 〈-〉키는 지정을 해제할 때에 사용한다. 그리고 현재의 디렉토리에 있는 모든 화일을 지정할 경우에는 〈Ctrl-F〉키를 사용한다(단, Type명령에서는 〈Ctrl-F〉키가 작동되지 않는다).

(3) Type

이 명령은 텍스트화일을 화면상에 나타내주는 명령이다.

(4) Rename

화일의 이름을 바꿀 때 사용한다. 1DIR에서는 디렉토리 이름도 화일이름으로 취급하므로 디렉토리이름도 이 명령으로 바꿀 수 있다.

(5) Erase

화일을 삭제할 때 사용한다. 이것을 실행시키면 화일의 삭제여부를 확인하는 메세지가 나오는 데, 여기에서 〈Y〉키를 누르면 그 화일을 삭제하고 다른 키를 누르면 화일삭제가 취소된다.

(6) Date

이것은 도스의 내부 명령 'DATE'과 같은 기능을 갖는다.

(7) MKDIR

새로운 서브디렉토리를 만들때 사용하는 명령이다. 이것을 실행시키면 서브디렉토리이름을 묻는데 여기에서 자신이 원하는 서브디렉토리이름을 입력하면 된다.

(8) Options

이것을 실행시키면 1DIR은 OPTIONS.EXE라는 프로그램을 실행시킨다. 이 옵션 프로그램은 1DIR의 각종 파라미터를 조정하는 프로그램이다.

배치 빌더

배치 빌더는 다른 도스 쉘 프로그램에서는 찾아보기 어려운 1DIR만의 독특한 기능이다. 이것은 1DIR의 수행과정을 하나의 배치화일로 담을 수 있게 해주는 기능으로서 배치화일을 많이 활용하는 유저에게는 특히 유용한 도구가 될 것이다.

배치 빌더의 실행순서는 다음과 같다.

① 코맨드 커서를 'Compose'에 둔다.
② 〈Ctrl-B〉를 눌러서 배치 빌더를 시작한다.
③ 배치화일의 이름을 지정한다. 지정한 화일이름으로 배치 빌더가 끝나면 배치화일이 생성되는 것이다. 여기서 화일이름을 입력할 때에는 확장명 'BAT'는 생략한다(이것은 1DIR이 자동적으로 붙여주기 때문에).
④ 일련의 코맨드를 준다. 여기에서 주는 코맨드는 모두 배치 빌더에 캡쳐되어서 나중에 ③에서 지정한 화일이름으로 저장된다.
⑤ 배치 빌더를 끝내려면 다시 〈Ctrl-B〉를 누른다. 이때 지금까지 캡쳐된 코맨드가 저장된 배치화일이 생성된다.

배치화일을 만드는 과정인 ④에서 실제로 코맨드를 실행시키지 않고서 캡쳐하기 위해서는 엔터키를 누르지 않고 〈Ctrl-R〉을 입력하면 된다.

사용키

1DIR에서 자주 사용되면서 유용한 키를 정리해 보았다.

〈Esc〉: 여러 스텝의 코맨드에서 중간에 취소할 때
〈Scroll Lock〉: 셋업모드로 들어가고 나갈 때 쓰이며 〈Ctrl-Q〉와 같다.
〈Ctrl-B〉: 배치 빌더의 시작과 끝
〈Ctrl-F〉: 디렉토리의 모든 화일을 지정
〈Ctrl-O〉: 모니터를 온/오프
〈Ctrl-\〉: 루트 디렉토리로 이동
〈Ctrl-C〉: 1DIR을 끝내고 도스로 나감
〈Shift-PrtSc〉: 프린팅 유틸리티를 호출

도스 쉘계의 거장 1DIR을 너무 장황하게 설명한 것 같다. 꼭 1DIR만이 가장 완벽한 유틸리티라고 확언할 수 없는 것은 실제 도스 쉘 프로그램들은 각기 나름대로의 특징이 있기 때문이다.

행렬을 이용한 이미지 변환

강영도/본지 취재부기자

컴퓨터 그래픽은 앞으로 계속 개발되고 응용될 유망한 분야이다. 주위를 보면 그래픽 관련 소프트웨어가 아니더라도 각종 응용프로그램에서 그래픽을 통한 사용자 인터페이스를 많이 볼 수 있을 것이다. 그만큼 시각적으로 금방 전달될 수 있는 장점을 갖고 있는게 사실이다. 이외에도 컴퓨터 그래픽을 이용하는 영역은 많고 또 이용될 수 있는 영역 또한 무한하다고 볼 수 있다.

다양한 그래픽 표현이 어떻게 표현되는지 궁금하기도 하고 자기가 직접 구현도 해보고 싶은 컴퓨터 그래픽의 입문자들은 대부분은 IBM PC BASIC 또는 GWBASIC을 가지고 그래픽을 익히는게 보통이다. 배우기 쉽고 컴퓨터 그래픽원리를 차례차례 익혀나가면서 배울수 있기 때문이다.

여기서는 화면의 특성을 고려한 윈도우 지정과 함께 2차원 배열을 이용하는 그래픽 이미지의 스케일링 및 위치변환 비틀기, 회전 등에 대한 기본 원리에 대해 알아보자.

우선 컴퓨터 화면이 갖는 기본적인 속성부터 알아보자. 컴퓨터의 화면 픽셀을 보면 가로와 세로가 서로 일정하지가 않다. 이것은 TV화면도 마찬가지인데 IBM PC의경우 CGA (Color Graphics Adaptor)를 기준으로 살펴보면 중해상도 그래픽 화면에서는 320*200, 고해상도 그래픽 화면에서는 640*200의 픽셀을 갖고 있기 때문에 당연히 가로와 세로의 픽셀 수가 틀리게 마련이다. 이 사실은 평상시는 중요하게 작용하지 않을 수 있지만 때로 가로와 세로로 정확한 (공학용)도형을 그릴 때는 문제가 될 수 있다. 이럴 때는 이미지를 그리기 전에 가로와 세로의 픽셀 비율을 재조정시켜 주어야 할 것이다. 즉 중해상도인 경우 모든 수직축은 5/8 (200/320)을 곱해서 나타내야 하고 고해상도 모드에서도 마찬가지 방법으로 재조정을 시켜야 한다. 그러나 이것만으로는 완전치가 못할 것이다. 가로 세로의 픽셀 수뿐만 아니라 화면자체의 종횡비(Aspect Ratio)도 함께 고려해야 하기 때문이다. 일반적으로 TV나 컴퓨터 화면의 종횡비는 4/3으로 되어 있다. 즉 화면의 폭은 높이의 4/3을 차지하고 있다는 말이 된다 (실제로 화면의 종횡비를 살펴보기 바란다).

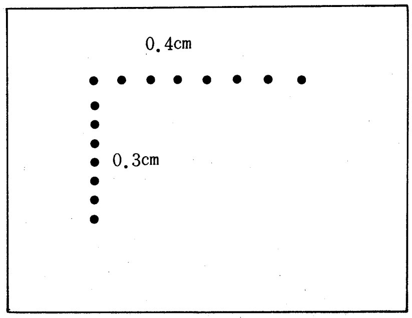
예를 들어 화면상에 가로로 8개의 픽셀을, 그리고 세로로 8개의 픽셀을 찍고 두 선분의 길이를 측정해 보면 비록 픽셀수는 같더라도 길이는 서로 다르다는 것을 알 수 있을 것이다(그림 1 참조). 즉 이 경우는 가로가 0.4cm, 세로는 0.3cm가 되어 세로축에 대한 가로축의 비율이 0.4/0.3, 즉 4/3가 되는 것이다. 이것을 화면이 갖는 종횡비(Aspect Ratio)라고 부른다.

이 5/8이라는 픽셀 비율과 4/3의 화면 종횡비를 곱하면 5/6이라는 스케일링 계수를 얻게 된다. 마찬가지 방법으로 고해상도 모드에서는 5/12이라는 계수를 얻을 수 있다. (국내 대부분의 PC사용자들이 사용하고 있는 허큘리스 보드에서는 CGA를 시뮬레이션 시키는 프로그램들을 이용하여 그래픽을 구현하고 있으며 이 경우는 720*348의 고해상도 화면을 얻을 수 있다) 아무튼 이같이 적절한 스케일링 계수를 도입하면 적당한 비율의 이미지를 얻을 수 있게 됨을 알 수 있다. 다음의 리스트를 보면 SCALE란 변수에 5/6을 지정한 후 도형의 Y축(VSIDE)을 재조정하고 있다.

```
10 SCALE = 5/6
20 SIDE = 100
30 VSIDE = SIDE * SCALE
40 X = 20: Y = 50
50 SCREEN 1: KEY OFF: CLS
60 LINE (X,Y) - STEP(SIDE, VSIDE),,B
```

그럼 매번 이런 불편한 재조정 작업을 해야하는가? 베이직 명령 중에 WINDOW문을 이용하면 이 문제를 쉽게 해결할 수 있다. 앞에서도 언급했듯이 보통은 일반 그래픽을

〔그림 1〕 픽셀로 살펴본 화면의 종횡비



주로 그리고 스케일 변환을 필요로 하는 그래픽 숙련자 외에는 이런 고급기능의 명령어를 자주 이용하지 못하고 있는 게 사실이다. 이 WINDOW명령은 사용자가 화면 경계좌표를 재정의할 수 있도록 해주는데 이같이 화면의 물리적인 제한을 벗어나 사용자가 임의로 지정해서 이용하는 공간을 윈도우라고 하며 이 좌표계를 세계좌표계(World Coordinate)라고 부른다 (**그림 2** 참조). 이에 비해 SCREEN 명령은 Y축을 역으로 하므로 X는 좌에서 우로 증가하고 Y는 위에서 아래로 증가하게 되는 것이다.

그렇다고 하더라도 WINDOW(0,0) - (120, 120) 으로 설정된 윈도우에서는 동일한 가로, 세로의 스케일을 갖지 못하고 대신 화면의 종횡비를 미리 고려해서 다음과 같이

```
WINDOW (0,0) - (120,90)
```

또는

```
WINDOW (0,0) - (160,120)
```

로 디스플레이 윈도우를 설정해 두면 구태여 화면의 종횡비 및 스케일링 계수 등을 고려하지 않아도 된다.

다음에 고려해야 할 것으로는 우리가 원하는 픽셀이 화면의 범위를 벗어났을 경우이다. 예를 들어 **그림 3**에서 사각형을 화면의 범위라고 했을 때 우리가 A에서 B까지를 잇는 선분, 즉 $\overline{AB}$를 그릴려고 하면 화면을 벗어나는 $\overline{CB}$ 부분은 그림과 같이 화면의 좌측으로 평행이동 되어 그려지게 된다 ($\overline{DE}$). 이런 현상을 랩효과 또는 래핑효과라고 부른다. 결론적으로 이렇게 지정된 화면의 범위를 벗어나는 선분이나 픽셀을 짤라버리거나 취소시켜 버리는 기능이 필요한데 이것을 클리핑(Clipping)이라고 한다. 특히 버전 2.0 이전의 IBM BASIC에서는 선분이 클리핑되었을 때 그 끝점의 좌표를 잃어버리는 단점이 있었으나 다행히도 요즈음의 베이직에서는 모든 클리핑 기능이 자동적으로 수행되며 끝점의 좌표 또한 계속 유지되므로 이 부분은 고려하지 않아도 문제가 없게 되었다.

이제부터는 각종 변환의 기초에 대해 알아보기로 하자. 이해를 돕기 위해 **리스트 1**의 예제를 보자.

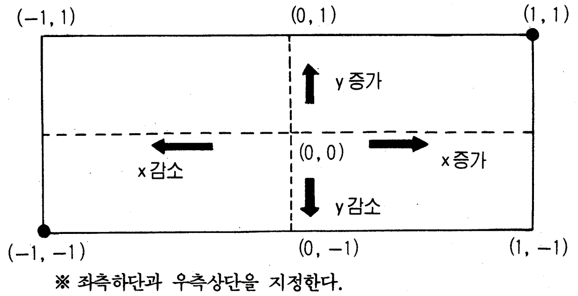
우선 이 프로그램을 살펴보면 DATA문으로 설정된 공통의 데이터를 변수 X, Y로 취한 후 화면좌표(SX, SY)를 변환시키면서 8개의 도형을 그리는데 여기서 도형의 크기와 모양, 그리고 위치는 A, B, C, D의 값에 의해 조절되고 있음을 볼 수 있을 것이다.

간단하게나마 이 프로그램은 다음과 같은 2종류의 변환을 보여주고 있다.

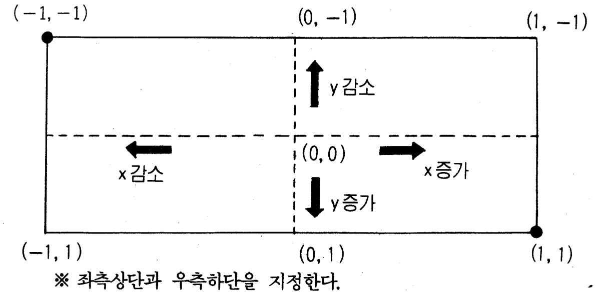
① A와 C가 곱해지면서 좌표의 크기가 제어되는데 이것을 스케일링(Scaling)이라고 한다.

② B와 D의 값에 의해 도형의 위치가 변환되고 있는데 이것을 이미지의 위치변환으로 볼 수 있다.

〔그림 2〕 세계좌표로 불리우는 사용자정의 좌표

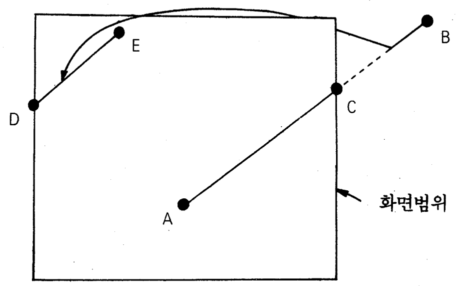


(a) WINDOW (−1, −1) − (1, 1)



(b) WINDOW SCREEN (−1, −1) − (1, 1)

〔그림 3〕 랩효과



〔리스트 1〕

```
10 SCREEN 2:KEY OFF: CLS
20 A = 20:B = 0:C = 10:D = 0
30 FOR J = 1 TO 8
40 READ X,Y:SX=A*X+B:SY=C*Y+D
50 PSET(SX,SY)
60 FOR I = 1 TO 4
70 READ X,Y:SX=A*X+B:SY=C*Y+D
80 LINE - (SX.SY)
90 NEXT I
100 A = A + 15:B = B + 15
110 C = C + 10:D = D + 10
120 RESTORE
130 NEXT J
140 LOCATE 23,1
150 DATA 0,0,1,0,1,1,0,1,0,0
```

스케일 변환과 위치변환을 좀더 구체적으로 표현하자면 리스트 1의 프로그램을 다음과 같이 변경하여 실행시켜 보면 알 수 있을 것이다.

① 8개의 도형 좌측 상단 꼭지점을 고정시키고 단지 크기만이 변하게 리스트 1의 프로그램을 변경해 본다. (스케일변환 또는 스케일링)

② 크기는 모두 같게 하고 위치만을 변화시키면서 8개의 도형을 그린다. (위치변환)

이러한 변환은 그래픽 표현에 매우 유용하게 쓰이는 기법으로, 특히 행렬 표현과 호모지니어스 좌표 표현에 의해 더욱 고급스러운 컴퓨터 그래픽에 이용되고 있는 소프트웨어 기술들이다.

행렬이란 다음과 같이 직사각형 배열로써 수가 나열되어 있는 형태를 말한다.

$$A=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix} \quad \text{또는} \quad B=\begin{pmatrix}0 & 2\\ 5 & -3\end{pmatrix}$$

이들 행렬의 덧셈은 다음과 같이 이루어 진다.

$$A+B=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}+\begin{pmatrix}0 & 2\\ 5 & -3\end{pmatrix}=\begin{pmatrix}1+0 & 2+2\\ 3+5 & 4-3\end{pmatrix}=\begin{pmatrix}1 & 4\\ 8 & 1\end{pmatrix}$$

한편 행렬의 곱셈을 살펴보면 다음과 같다.

$$\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}\begin{pmatrix}0 & 2\\ 5 & -3\end{pmatrix}=\begin{pmatrix}(1)(0)+(2)(5) & (1)(2)+(2)(-3)\\ (3)(0)+(4)(5) & (3)(2)+(4)(-3)\end{pmatrix}$$
$$=\begin{pmatrix}10 & -4\\ 20 & -6\end{pmatrix}$$

느닷없이 행렬이 등장해 의아해 하는 독자도 있을 것이다. 행렬은 그래픽 응용의 대부분의 분야에서 쓰이고 있으며 그 공식 또한 수없이 개발되고 있는 실정이다. 그렇지만 대부분의 어려운 행렬공식들은 수학자들에 의해 만들어지고 그래픽 디자이너들은 그 공식만 이용해도 되는 것이다. 그리고 여기서는 그래픽 구현원리만 이해하면 되므로 가장 기초적인 내용만을 다룬다 (**그림 4** 참조).

변환의 행렬표현

2차원 평면(컴퓨터화면)에서의 점들의 변환은 2×2행렬과의 곱(그 결과는 1×2행렬이 됨)으로써 표현될 수 있다. 한 예로 좌표(X, Y)의 점을 (X′, Y′)로 이동시키기 위해서 다음과 같은 수식을 이용한다.

$$(X, Y)\begin{pmatrix}A & B\\ C & D\end{pmatrix}=(AX+CY, BX+DY)=(X', Y')$$

즉 각 점 (X, Y)의 변환된 점은 다음과 같이 표현된다.

$$(X,Y)'=(X',Y')=(AX+CY, BX+DY)$$

여기까지 보면 변환된 효과는 A, B, C, D의 값에 의해 결정된다는 것을 알 수 있을 것이다. 그럼 실예로써 사각형을 구성하는 픽셀들이 변환되는 모습을 살펴보자. (**리스트2참조**)

리스트 2를 보면 40번에서 70번까지는 A,…,D까지의 값을 받아들이는 곳이고 220번에서 280번까지는 선택 메시지와 데이터문이다. 실제로 눈여겨 볼 곳은 120번부터 시작되는 루프문의

```
TX = A*X + C*Y
Y  = B*X + D*Y
```

로서 앞에서도 설명한대로 (X, Y)와 A, B, C, D로 이루어진 행렬을 곱하는 곳이다. 그럼 지금부터 이 프로그램을 가지고 각종 변환을 차례로 알아보자.

행렬

$$\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$$

은 아무런 변화가 없는 그대로의 변환을 나타낸다. 즉

$$(X,Y)\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}=(X,Y)$$

로서 자기 자신을 표현하게 된다. 리스트 2을 실행시켜 A=1,

〔그림 4〕

$$A=\begin{pmatrix}a_{11} & a_{12} & a_{13} & \cdots\\ a_{21} & a_{22} & a_{23} & \cdots\\ a_{31} & a_{32} & a_{33} & \cdots\\ \vdots & \vdots & \vdots & \end{pmatrix}$$

※ 밑에 붙은 첨자는 행과 열번호를 나타낸다.

(a) 행렬표현

$$\underset{A}{\begin{pmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{pmatrix}}+\underset{B}{\begin{pmatrix}b_{11} & b_{12}\\ b_{21} & b_{22}\end{pmatrix}}=\underset{C}{\begin{pmatrix}a_{11}+b_{11} & a_{12}+b_{12}\\ a_{21}+b_{21} & a_{22}+b_{22}\end{pmatrix}}$$

※ 같은 요소끼리 더한다.

(b) 행렬 덧셈

$$\underset{A}{\begin{pmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{pmatrix}}\underset{B}{\begin{pmatrix}b_{11} & b_{12}\\ b_{21} & b_{22}\end{pmatrix}}=\underset{C}{\begin{pmatrix}a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22}\\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22}\end{pmatrix}}$$

※ A의 첫행과 B의 첫열을 대입하여 C의 첫요소를 만든다.
(A의 행수와 B의 열수가 같아야 됨을 알 수 있다.)

(c) 행렬 곱셈

B=0, C=0, D=1의 값을 입력하면 아무런 변화가 없는 일정한 사각형 도형을 얻게 될것이다(화면 1 참조).

이번에는 스케일 변환을 해보자. 스케일 변환은 다음과 같은 대각행렬로써 표현되어 진다. 즉

〔리스트 2〕

```
10 SCREEN 2: KEY OFF: CLS
20 WINDOW (-4,-3) - (4,3)
30 '-----------------------------------------------------------
40 LOCATE 24,1: INPUT; "A =";A
50 LOCATE 24,15:INPUT; "B =";B
60 LOCATE 25,1: INPUT; "C =";C
70 LOCATE 25,15:INPUT; "D =";D
80 '-----------------------------------------------------------
90  LINE (0,3) - (0,-3)
100 LINE (3,0) - (-3,0)
110 '-----------------------------------------------------------
120 FOR LIN = 1 TO 4
130     READ X,Y
140     TX = A*X + C*Y : Y = B*X + D*Y
150     X = TX
160     PSET (X,Y)
170     READ X,Y
180     TX = A*X + C*Y : Y = B*X + D*Y
190     LINE - (X,Y)
200 NEXT LIN
210 '-----------------------------------------------------------
220 RESTORE: LOCATE 23,1
230 INPUT "새로운 행렬을 얻을려면 'N'을, 끝내려면 'E'를 누르시오";ST$
240 IF ST$ = "n" OR ST$ = "N" THEN CLS: GOTO 40
250 IF ST$ = "e" OR ST$ = "E" THEN CLS: END
260 GOTO 230
270 DATA 0,0, 1,0, 1,0, 1,1
280 DATA 1,1, 0,1, 0,1, 0,0
```

$$(X,Y)\begin{pmatrix} A & 0 \\ 0 & D \end{pmatrix} = (AX, DY)$$

기존의 좌표, (X, Y)에 각각 A와 D가 곱해져 그만큼 스케일되는 것이다.

예를 들면 X축 방향으로 2배만큼 확대하고 싶으면, 다음과 같이 행렬을 재지정한다.

$$\begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix}$$

즉 A=2, B=0, C=0, D=1을 입력하고 프로그램을 실행시키면 원하는 결과를 얻게 된다(화면 2 참조).

이번에는 Y축에 대해 거꾸로 스케일링시켜 도형을 수축시켜 보자. 다음과 같이 Y축에 1/5란 인자를 곱하면 될 것이다(화면 3 참조).

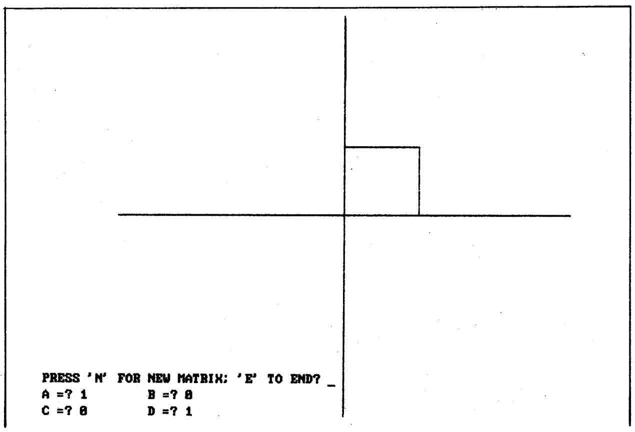
$$\begin{pmatrix} 1 & 0 \\ 0.2 \end{pmatrix}$$

반사효과

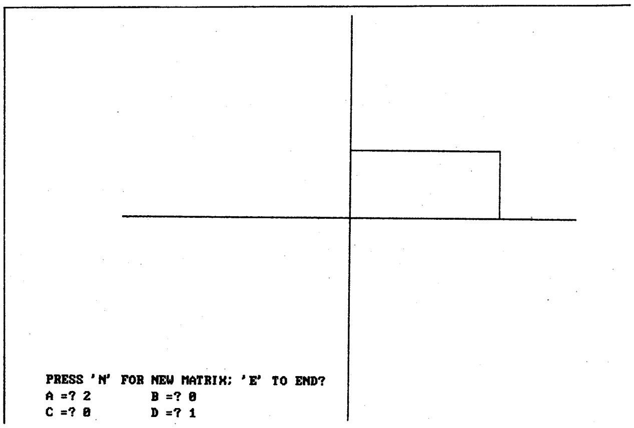
이미지의 반사에 대해서 알아보자. 이것은 기존의 이미지가 마치 거울에 비치는 것처럼 Y축이나 X축, 또는원점(0,0)을 중심으로 이미지를 반사시키는 것이다.

Y축을 중심으로 반사시키기 위해서는 X의 값을 음수로 만들어 주어야 하므로 A의 값을 음수로 잡아주고, 반대로 X축을 중심으로 반사시키기 위해서는 Y값을 음수 (더 정확히 표현하자면 기존의 Y값과 부호가 반대인 수)로 만들어 주어야 하므로 D의 값을 음수로 잡아주면 된다. 이 사실을 토대로 보면 원점(0,0)을 중심으로 반사시키는 것은 A와 D의 값을 반대부호의 수로 대치하면 된다는 것을 금방 알 수 있다(그림 5 참조).

〔화면 1〕



〔화면 2〕



실제로 A와 D의 값을 변화시키면서 위의 결과를 관찰해 보기 바란다.

비틀기 효과

다음의 행렬

$$\begin{pmatrix} 1 & 1.5 \\ 0 & 1 \end{pmatrix}$$

곧 A=1, B=1.5, C=0, D=1을 입력하여 프로그램을 실행시켜 보자. 출력을 보면 45° 방향으로 비틀어진 도형으로 변했음을 볼 수 있다(화면 4 참조). 이 행렬을 대수학적으로 살펴보면

$$(X', Y') = (X, Y) \begin{pmatrix} 1 & 1.5 \\ 0 & 1 \end{pmatrix} = (X, 1.5X+Y)$$

로 결국 X축의 좌표는 고정시킨 상태에서 Y축의 좌표를 1.5X 만큼 더하게 되어 비틀어진 도형이 된 것이다.

회전 효과

도형을 원하는 각도로 회전시켜 보자. 일반적으로 이미지 픽셀을 회전시키기 위해서는 다음과 같은 행렬을 이용한다.

$$\begin{pmatrix} \text{COS}\,\theta & \text{SIN}\,\theta \\ -\text{SIN}\,\theta & \text{COS}\,\theta \end{pmatrix}^{*}$$

지금까지의 스케일 변환과 반사, 비틀기 등에 관한 행렬식은 매우 간단한 표현이었지만 회전에 관한 행렬은 비교적 어렵게 느껴질 지 모르겠다.
이해를 돕기 위해 다음의 예를 보기로 하자.

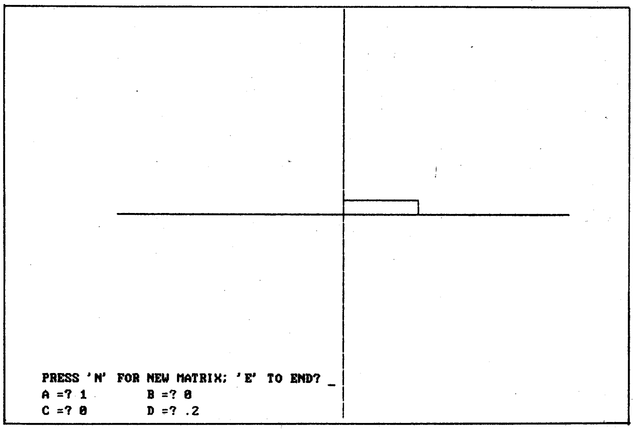
$$(A, B) = (1, 0) \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

그리고

$$(C, D) = (0, 1) \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

즉 위의 행렬변환을 보면 행렬의 첫번째 줄(A, B)는 점(1,0)에서의 변환으로 결정되며 두번째 줄의 (C, D) 는 점 (0,1)에서의 변환으로 결정되고 있음을 알 수 있다. 이제는 도형을 회전시키기 위해서는 이 두점, (1, 0) 와 (0,1) 에서 변환을 시켜주면 된다는 사실을 알게 되었는데 단지 차이점은 이 두점이 일정한 정수가 되는 것이 아니라 임의의 각도로써 표현된다는 것 뿐이다. 그럼 임의의 각 θ를 갖는 지점의 좌표는 어떻게 구할까? 이럴때 바로 삼각함수가 필요하게 된다.

〔화면 3〕



〔그림 5〕 반사효과

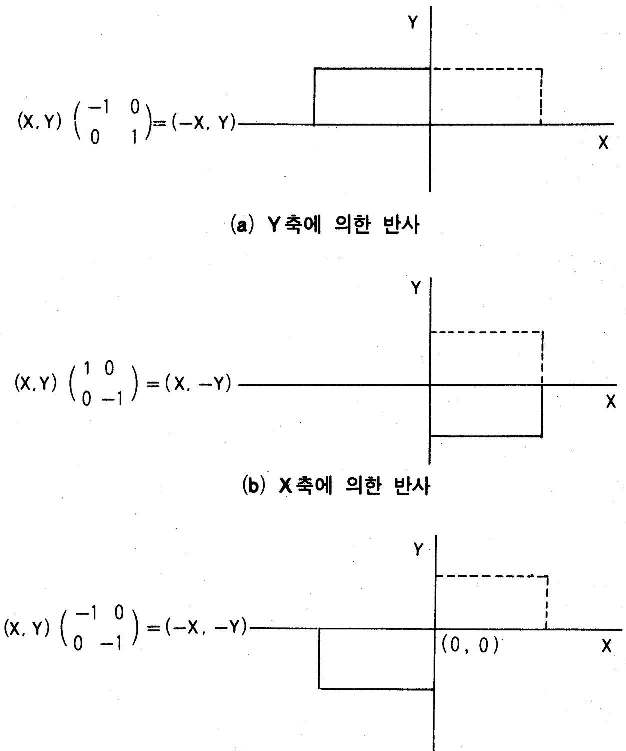
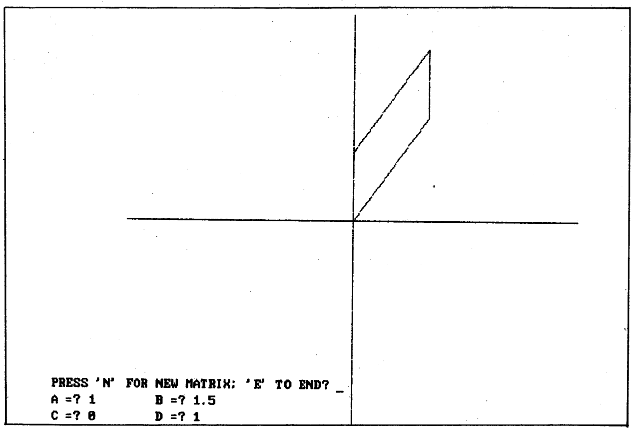


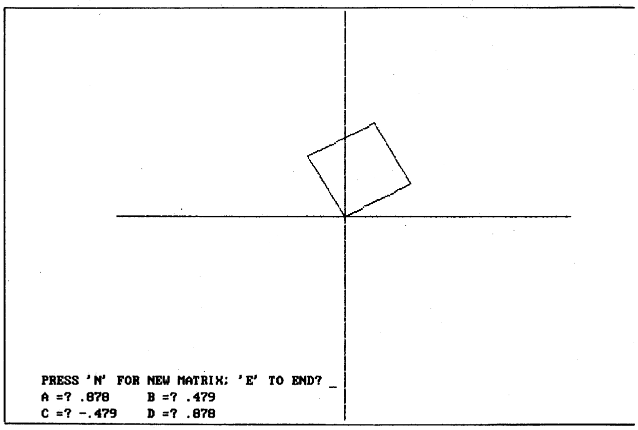
그림 6에서 (b)를 보면 사인과 코사인 함수에 의해서 각 좌표(좀더 정확히 표현하면 원점을 기준으로 X축과 Y축에서의 길이)를 구하는 원리를 알 수 있다. 여기서는 선분의 길

* θ란 기호는 그리이스 문자로 'THETA'라 발음되며 보통 각도를 표현할 때 쓰인다. 편의상 여기서는 베이직을 예로 설명되고 있기 때문에 이 θ를 TH또는 THETA로 표현하였다. 그리고 삼각함수는 베이직에 있는 예약어를 이용해야 되기 때문에 SIN(TH)와 같이 괄호를 첨가했다.

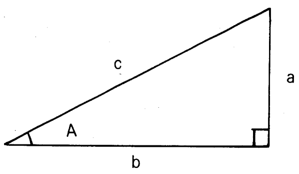
〔화면 4〕



〔화면 5〕



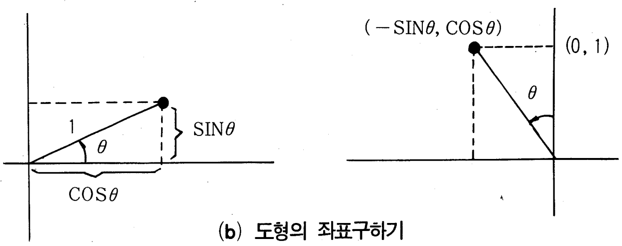
〔그림 6〕



$$\mathrm{SIN}\,(A) = \frac{a}{c}$$

$$\mathrm{COS}\,(A) = \frac{b}{c}$$

(a) 삼각함수 구하기



(b) 도형의 좌표구하기

이를 1로 하고 있으나 이 길이는 임의의 길이라도 괜찮다. 왜냐하면 그 선분의 길이만큼 삼각함수에 곱해주기만 하면 되기 때문이다.

종합해 보면 삼각함수에 의해 시계 반대방향으로 각 θ를 갖는 점(COS θ, SIN θ)에서 (1,0)의 회전을 얻을 수 있고, 같은 각을 취하는 점(−SIN θ, COS θ)을 통해 (0,1)의 회전을 얻을 수 있음을 알게 된다. 결국 회전에 이용될 수 있는 행렬은 다음과 같이 요약된다.

$$\begin{pmatrix} \mathrm{COS}\,\theta & \mathrm{SIN}\,\theta \\ -\mathrm{SIN}\,\theta & \mathrm{COS}\,\theta \end{pmatrix}$$

화면 5는 A=COS(.5)=.878, B=SIN(.5)=.479, C=−SIN(.5)=−.479, D=COS(.5)=.878란 데이터를 입력하여 프로그램을 실행시킨 결과를 보인 것이다. **

이제는 회전된 이미지를 가지고 앞에서 다루었던 스케일 변환과 반사 및 비틀기를 쉽게 표현할 수 있게 되었다. 예를 들어 반시계 방향으로 θ=.4라디안의 각도로 회전되고 X축이 인수 2로 스케일링 되는 변환을 정의해 보자.

우선 회전은 다음과 같이 이루어진다.

$$(X',Y') = (X,Y)\begin{pmatrix} \mathrm{COS}\,(.4) & \mathrm{SIN}\,(.4) \\ -\mathrm{SIN}\,(.4) & \mathrm{COS}\,(.4) \end{pmatrix}$$

그리고 스케일 변환은 다음과 같이 된다.

$$(X'',Y'') = (X',Y')\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

여기서 (X', Y')를 대입하면,

$$(X'',Y'') = \left[(X,Y)\begin{pmatrix} \mathrm{COS}\,(.4) & \mathrm{SIN}\,(.4) \\ -\mathrm{SIN}\,(.4) & \mathrm{COS}\,(.4) \end{pmatrix} \right]\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

$$(X'',Y'') = (X,Y)\left[\begin{pmatrix} \mathrm{COS}\,(.4) & \mathrm{SIN}\,(.4) \\ -\mathrm{SIN}\,(.4) & \mathrm{COS}\,(.4) \end{pmatrix}\begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \right]$$

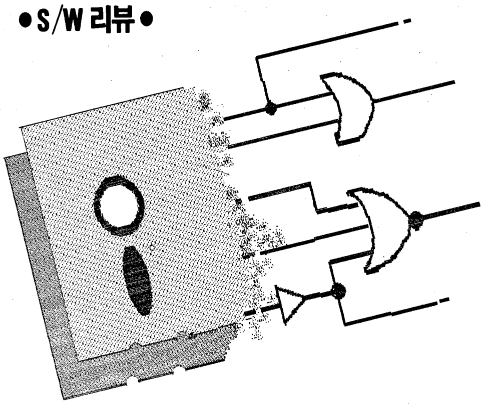
$$= (X,Y)\begin{pmatrix} 2\mathrm{COS}\,(.4) & \mathrm{SIN}\,(.4) \\ -2\mathrm{SIN}\,(.4) & \mathrm{COS}\,(.4) \end{pmatrix}$$

가 되어 최종적으로 변환된 행렬은 다음과 같이 된다.

$$\begin{pmatrix} 2\mathrm{COS}\,(.4) & \mathrm{SIN}\,(.4) \\ -2\mathrm{SIN}\,(.4) & \mathrm{COS}\,(.4) \end{pmatrix} = \begin{pmatrix} 1.842 & .389 \\ -.779 & .921 \end{pmatrix}$$

지금까지 2차원 행렬을 이용한 여러 종류의 변환을 살펴보았다. PC

** 베이직에서는 각을 라디안 값으로 받아들인다. 1라디안은 약 57.3°(디그리 값)를 가지는데 정확한 표현은 2π라디안이 360°이므로 1라디안 값은 360°/6.2832=57.2956°이 된다.



회로해석이 가능한 소프트웨어
마이크로 로직

박일수 / 중앙대학교 전자공학과

서문

컴퓨터 관련학과(특히 전자계산학과, 전자공학과)에서 대개 2학년 교과과정 중 필수적으로 배우는 것이 '디지탈공학'이다. 이 디지탈공학에서는 각종 게이트로 로직을 설계하고 분석하는 방법을 배운다. 본지(퍼스널컴퓨터)에서 창간호부터 연재한 '디지탈강좌'를 애독한 독자라면 처음은 쉬웠으나 게이트가 3~4개 조합된 플립플롭 등의 해석은 매우 어렵다는 것을 느꼈을 것이다.

논리회로를 해석해 주는 대표적인 프로그램 중의 하나가 OR CAD인데 프로그램 사용법이 다소 복잡하다는 점 때문에 그리 폭넓게 쓰이는 것 같지는 않다. 독자들이 벌써 눈치챘을지도 모르겠지만 여기서 소개하려는 마이크로 로직(MICRO LOGIC)은 필자가 지난 3월호에 소개한 마이크로 캡(Ⅱ)과 한 회사의 제품으로 프로그램의 전체적인 구성이 비슷하다. 그러나 사용법은 상당히 다르며 앞으로 설명 과정에서 나타나겠지만 마이크로 캡(Ⅱ)에 비해 다소 비효율적으로 동작하는 부분도 있다.

아미크로 캡(Ⅱ)에서 하드카피를 해 본 독자는 알겠지만 마이크로 로직도 캐드(CAD)용으로 만족스러운 그래픽 기능을 갖추고 있으며 예쁜(?) 하드카피를 얻을 수 있다. 그리고 마이크로 로직 또한 APPLE Ⅱe에서도 실행이 가능하므로 8비트 사용자들도 사용해 보기 바란다.

그리고 마이크로 로직의 사용법은 메뉴만 많을 뿐이지 쉽게 배울 수 있으므로 본문에서 나오는 4개의 예제를 잘 이해해 주었으면 한다. 본문의 예제는 '디지탈 강좌'의 예제를 그대로 사용하려고 하니 비전공자들 중 논리회로를 공부하려는 사람은 과월호의 연재 내용을 보면서 마이크로 로직을 배우면 더욱 효과적일 것이다.

본문

(1) 셋업

우선 동작에 필요한 하드웨어를 보면 다음과 같다.

• IBM PC의 경우

① 512KB 이상의 RAM
② 2대의 FDD
③ 모노 또는 컬러 그래픽 카드
④ 프린터

• APPLE의 경우

① APPLE Ⅱ PLUS 또는 Ⅱe
② 64K RAM
③ 2대의 FDD
④ 프린터

• 인스톨

마이크로 로직은 프로그램 디스켓 1장과 데이터 디스켓 1장으로 구성되어 있다. 작업을 할 때에는 백업본을 만들어 사용하고 원본 디스켓은 안전한 장소에 보관해 둔다. 또한 실행하기 전에 하드카피를 대비해서 GRAPHICS.COM을 실행시키고 허큘리스 보드를 사용하는 독자는 SIMCGA1 화일도 같이 올려 사용해야 한다. 이 과정을 다음과 같이 배치화일로 만들어 놓고 사용하면 더 편리해 진다.

① 마이크로 로직의 프로그램 디스켓을 A드라이브에 넣고 데이터 디스켓을 B드라이브에 넣는다.
② A프롬프트 상에서 다음과 같이 입력한다.

```
A:\>COPY CON AUTOEXEC.BAT
GRAPHICS
SIMCGA1
INTRO
^Z
```

마이크로 로직은 실행시 반드시 A드라이브에는 프로그램 디스켓이, B드라이브에는 데이터 디스켓이 들어 있어야 함을 잊지 말기 바란다.(내부적으로 드라이브 A와 B로 세트되어 있기 때문에 이를 수정하지 않는 한 당분간은 하드디스크에서의 실행이 불가능하게 되어 있다)

(2) 프로그램의 구성

마이크로 로직의 구성은 **그림 1**에서 나타낸 것처럼 4개의 모듈로 되어 있으나 디자인모듈과 분석모듈은 또 각각 4~8개의 모듈로 분리되어 프로그램의 전체 구성은 비교적 복잡하며 이에 따른 선택 옵션도 매우 많다.

(3) 게이트의 설계

일단 디스크 1, 2를 드라이브에 넣고 프로그램을 실행시켜 보자. **화면1**을 볼 수 있을 것이다. 여기서 디자인모듈 1을 선택한다. 그러면 다시 다음과 같은

메뉴가 나온다.

```
1.Initialize Data Diskette
2.Run The program
3.Edit/Review GATE Shapes
4.Return to main menu

            Enter your choice number
```

이 메뉴에서 3번을 선택하면 **화면 2**와 같이 약 20개 정도의 게이트 및 플립플롭들이 보인다. 앞으로의 작업을 위해 게이트의 이름들을 주의깊게 살펴봐 주기 바란다.

마이크로 로직으로 논리회로를 구성할 때 게이트, 클럭, 채널, 3단 게이트, 데이터 등의 사용갯수에는 제한이 있다.

그것을 **표 1**에 나타내었으므로 참고하기 바란다.

아마 어느정도 복잡한 회로를 구성했을지라도 표 1 정도라면 독자들은 거의 모자람이 없이 회로를 구성하여 사용할 수 있을 것이다.

(4) 디자인모듈

우선 원하는 논리회로를 구성하고 수정할 수 있는 디자인모듈에 대해서 설명한 후 몇가지 간단한 예를 다루어 보기로 하겠다.

• 데이터디스켓의 초기화

메뉴에서 1을 눌러 이 옵션을 선택하면 다음의 질문이 나온다(보기 1)

'Y'를 누르면 또 다음의 메시지가 나오는데 독자들은 역시 'Y'를 누르기를 바란다. (보기 2)

이 상태에서 아무키나 누르면 드라이브가 구동되면서 초기화가 시작된다. 여기서의 초기화는 디스크의 초기화가

〔화면 1〕 디스크 메뉴

```
        SPECTRUM  SOFTWARE
MICRO-LOGIC REV 2.5...08/01/84
        COPYRIGHT  1983,1984

            DISK MENU

   1 : RUN DESIGNER MODULE

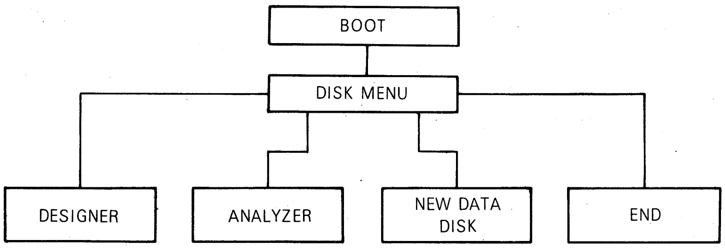
   2 : RUN ANALYZER MODULE

   3 : MAKE A NEW DATA DISK

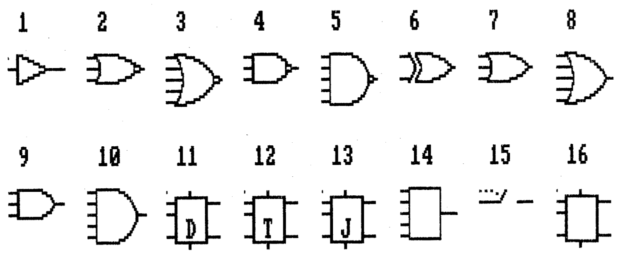
   4 : END THE PROGRAM

  ENTER YOUR CHOICE BY NUMBER
```

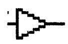
〔그림 1〕 마이크로 로직의 구성



〔화면 2〕 정의되어진 게이트 및 플립플롭



GATE # 1 'S NAME IS INV AND IT'S SHAPE IS # 1 ...



COMMANDS: N=NEXT GATE   L=LAST GATE   E=EDIT SHAPE # USED   Q=QUIT ROUTINE

〔표 1〕 마이크로 로직이 갖는 최대 게이트, 클럭, 채널등의 사용 갯수

사양(최대 갯수)	APPLE	IBM
#of types	18	36
#of data channels	18	36
#of clocks	9	9
#of MACROS in a netlist	25	50
#of total gates	432	1750
#of states in a data channel pattern	256	256
Delay of any gate-calculated	13	13
Delay of any gate-fixed with variable	9	9
Resolution(#of sub-cycles per clock)	50	100
#of Tri-state gates	50	200
#of Labels	70	200

〔보기 1〕

```
THIS ROUTINE ERASES ANY DESIGNER, DATA FILE ON THE TARGET
DATA DISKETTE IN DRIVE B: DO YOU WISH TO CONTINUE(T/N) ?
```

〔보기 2〕

```
PLEASE INSERT DATA DISK INTO DRIVE B. THE DISK SHOULD HAVE
BEEN FORMATTED USING THE DOS FORMAT COMMAND... FORMAT B:..
PRESS ANY KEY TO START OR PRESS Q TO QUIT WITHOUT INITIALIZING
```

아니라 데이터의 초기화를 의미하므로 라이브러리나 매크로, 패턴 등의 화일은 영향을 받지 않는다. 즉 초기화는 새로운 회로를 그리기 위한 디자인모듈의 청소작업이라 하겠다. 그러므로 독자들은 새로운 논리회로를 그릴때 이 과정을 반드시 거쳐야 한다는 것을 알아 두자.

• 프로그램 실행

옵션 2를 선택하면 다음의 메시지가 나온다.

```
Which Page(0-8)?
```

뒤에 설명하겠지만 마이크로 로직은 9개의 페이지를 가지고 있는데 항상 0페이지가 우선적으로 선택되어야 한다. 독자들은 0을 치고 리턴을 누르거나 또는 그냥 리턴키만 눌러주면 회로를 디자인할 수 있는 에디터 상태로 넘어가게 된다.

우선 H를 눌러보면 화면 상단에 에디터를 사용하는 방식 및 키 사용법이 간략히 나온다. 그러나 주의할 점이 몇가지 있으므로 필자가 다시 한번 정리해 본다. 특히 커서 이동은 I, M, J, K 키가 각각 상하좌우키로 동작하고 U, O, N, ‘,’키가 각각 대각선방향의 키로 동작한다.

① L : 로컬의 의미를 가지며 그 커서가 위치한 영역속의 게이트의 출력 단자에 커서가 위치하게 된다.

② F : 커서의 이동간격을 조절해준다. (1X와 4X가 있다)

③ A : 화면에 게이트를 선택하여 그리는 기능을 담당한다. 이것을 선택하면 몇가지 질문이 나오는데 이것은 앞으로 소개할 예제를 통해 쉽게 익힐 수 있다.

④ R : 잘못 그려진 게이트를 지우는 기능을 가진다.

⑤ C : 연결의 의미를 가지며 조심해서 사용해야 할 기능이다. 마이크로 로직에서는 선이 한점에서 교차해도 C에 의한 노드가 생기지 않으면 접속되지 않은 것으로 간주한다.

⑥ P : 펜을 뜻한다. UP, ERASE, DRAW의 세가지 상태를 갖는다. 토글방식에 의해 UP와 ERASE는 교차되며, DRAW는 C에 의해 나타나게 된다. 예제에서 확실한 것을 알게 될 것이다.

⑦ H : 도움말의 내용을 화면의 상단에 보여준다.

⑧ Q : 편집기능을 끝내며 현재 페이지의 형태를 ‘SIMFILE’이라는 이름으로 저장한다.

⑨ S : 페이지를 바꾸어준다. 다음 질문을 묻는데 원하는 페이지 번호를 입력해 주면 된다.

⑩ B : 화면을 하드카피하기 전에 주로 쓰는 기능인데 화면상단의 메뉴를 없애버린다.

• 게이트의 종류와 모양을 보여주기

화면 2를 참조하여 독자들이 각 게이트의 형태와 이름을 기억하기 바란다. 주의깊게 본 독자들중 왜 NAND, AND 등의 게이트에 입력단자가 2개인 것이 없냐고 질문하는 사람이 있을 수 있는데 그것은 3개의 입력단자중 2개를 C로 붙여서 사용하면 되므로 회로의 모양은 약간 이상할지 모르나 기능은 같은 것이 된다.

〔표 2〕 키보드로 입력해야 할 게이트 이름

게 이 트	키보드로 입력되는 이름
인버터	INV
3입력 NOR	NOR 3
5입력 NOR	NOR 5
3입력 NAND	NAND 3
5입력 NAND	NAND 5
exclusive-OR	EX-OR
3입력 OR	OR3
5입력 OR	OR5
3입력 AND	AND3
5입력 AND	AND5
D-플립플롭	DFFTTL
T-플립플롭	TFFTTL
JK-플립플롭	JKFFTTL
버퍼	BUFFER
3-state-gate	TGATE

• 메인 메뉴로 돌아가기

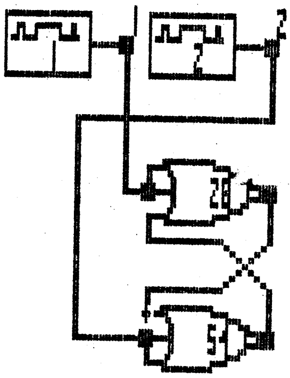
메인메뉴(부팅후 처음메뉴)로 돌아가 다음명령을 기다린다.

디자인 모듈에서 한가지 더 알아야 할 사항이 있다면 각 게이트의 입력핀 번호이다. 인버터인 경우는 입력핀이 하나이므로 핀번호는 1이 된다. Exclusive OR나 T게이트인 경우는 왼쪽 위부터 차례로 1, 2번이 되고 3 또는 5입력 게이트는 역시 왼쪽 위부터 차례로 1~5번이 된다. 마지막으로 플립플롭 게이트인 경우는 왼쪽 위부터 1, 2, 3번이고 게이트 상단의 핀이 4번, 하단의 핀이 5번이다.

(5) 예제

예제에 들어가기 전에 원하는 게이트를 그리기 위해 키보드로 입력해야 할 각 게이트 이름을 미리 익혀두면 좋을 것이다(표 2 참조). 입력핀에 인가하는 펄스형 데이터와 클럭은 다음

〔그림 2〕 RS 플립플롭



과 같다.

입력 데이터→DATA 1
DATA 2
⋮
클 력→PHASE 0
PHASE 1
⋮

• 예제 1

3월호 '디지탈강좌'에 나오는 2개의 NOR 게이트로 구성된 RS 플립플롭을 구성해 보자. 그림 2에 이미 나타나 있는데 첫 예제이기 때문에 자세히 설명하려고 하므로 필자가 제시하는 방법 그대로 해주기 바란다. 디스크 메뉴에서 다음 순서로 입력을 한다.

1 —— 디자인모듈을 선택
2 —— 프로그램 실행 메뉴 선택
0 ↵ 0 페이지를 선택

A
DATA 1 ↵
RIN ↵] 레이블이 RIN인 데이터 입력

K —— 커서를 우측으로

A
DATA 2 ↵
SIN ↵] 레이블이 SIN인 데이터 입력

M 커서를 아랫쪽으로
M 커서를 아랫쪽으로

A
NOR3 ↵
QP ↵] 레이블이 QP인 NOR게이트를 그린다.

M
M

A
NOR3 ↵
QN ↵] 레이블이 QN인 NOR게이트를 그린다.

데이터 입력기와 게이트는 모두 그렸다. 이제 각 핀과 핀을 연결해 보기로 하자.

I —— 커서를 위로 올린다.
I
I
I

F —— 커서의 이동간격을 X4에서 X1으로 바꾼다.

J
M] DATA1의 출력핀에 커서가 위치해 있을 것이다.

C —— 연결 기능을 선택
F —— 다시 X4로 바꾼다.
M

M
F
K
C] 선이 그려지면 QP게이트 가운데(pin2)에 연결

C
I
C] QP의 핀1과 핀2를 연결

참고로 여기서 C는 C로 끝나야 한다는 것을 알아두자. 그리고 C를 선택하면 PEN : DRAW로 상단에 나타나는 것도 주의깊게 보기 바란다. 여기까지 그리는 도중에 잘못그린 독자는 P를 눌러보기 바란다.

PEN : ERASE로 바뀌었을 것이다. 잘못된 곳을 다 지운후 다시 P를 누르면 PEN : UP상태로 돌아온다. 그러면 계속해서 나머지 회로를 완성해 보도록 하자.

F
K
I
F
I
I
J
C] Data2의 출력핀에 접촉시킨다.

F
M
J
J
M
M
M
K
F
K
C] DATA 2의 출력과 QN의 핀 2가 연결되었다. 앞에서 언급한 바와 같이 선이 교차하더라도 무방하다.

C
M
C] QN의 핀 2와 핀 3을 연결시킨다.

L 커서가 QN의 출력핀쪽으로 바로 옮겨진다.

'L'의 기능으로 여러번 키조작을 하지 않아도 된다.

C
I
U
U
U
J
J
I
C] QN의 출력과 QP의 핀3이 연결되었다.

L 커서가 QP의 출력쪽으로 이동한다.

C
M
N
N
N
J
J
M
C] QP의 출력과 QN의 핀 1이 연결되었다.

회로가 그림 2처럼 완성되었을 것이다. Q를 눌러 회로를 저장시키고 이번에는 분석모듈로 가보기로 하자.

Q —— 화일에 회로의 형태와 특성이 저장된다.
2 —— 분석시작 모듈 선택

2번을 선택하면 1~8까지의 분석모듈 메뉴가 나타났을 것이다. 이것의 각 기능은 뒤에 자세히 소개하겠으니 일단 의문사항없이 필자의 말대로 해주기 바란다.

2 ← 디스크 작업 선택

이번엔 화면에 디스크 작업에 관한 7개의 선택메뉴가 나타날 것이다. 이 상태에서 다음과 같이 입력해 보자.

3 ← 회로도 복원 선택

↵ ─ 그냥 리턴을 한다. 컴퓨터가 자동으로 SIMFILE을 참조하여 회로를 구성해 준다.

7 ← 분석 메뉴로 돌아온다.

1 ─ 화면 3이 나타났을 것이다. 우리가 정의한 2개의 데이터 패턴이 레이블과 함께 나타난다.

Q ← 분석 메뉴로 돌아온다.

우리는 2개의 데이터 패턴의 이름만 지정했지 값은 정의해주지 않았다. 그럼 이번에는 5를 선택하여 **화면 4**의 상태에서 값을 정의해 보자.

E
1 ↵ } 1번을 수정한다.

0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 ↵ ← 값 입력

E
2 ↵ } 2번을 연결한다.

1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 ↵ ← 값 입력

Q ← 분석 메뉴로 돌아온다.

위의 데이터는 필자가 임의로 정한 것이며 DATA1과 DATA2가 동시에 1인 경우는 RS 플립플롭에서는 정의되지 않은 상태이므로 제외시켰다. 동시에 1인 경우가 궁금한 독자는 아래 필자가 제시하는 시뮬레이션 후에 패턴의 값을 변화시켜 다시해 보기 바란다.

3 ← 시뮬레이션 옵션을 선택

이번에는 13개의 옵션을 묻는 프롤로그 화면이 나타났을 것이다. 자세한 것은 역시 뒤에 설명하겠으니 다음과

〔화면 3〕 데이터 패턴

```
NO. NAME     LABEL      A      B      C      D      E      F      G      H
1   DATA 1   RIN        0      0      0      0      0
2   DATA 2   SIN        0      0      0      0      0
3                       0      0      0      0      0
4                       0      0      0      0      0
5                       0      0      0      0      0
6                       0      0      0      0      0
7                       0      0      0      0      0
8                       0      0      0      0      0
9                       0      0      0      0      0
10                      0      0      0      0      0
11                      0      0      0      0      0
12                      0      0      0      0      0
13                      0      0      0      0      0
14                      0      0      0      0      0
15                      0      0      0      0      0
16                      0      0      0      0      0
17                      0      0      0      0      0
18                      0      0      0      0      0
19                      0      0      0      0      0
20                      0      0      0      0      0
UP  DOWN  LEFT  RIGHT  JUMP  ADD  EDIT  HELP  ZAP  CLEAR  PRINT  QUIT
CURRENT MACRO COUNT =  0  MAXIMUM =  50
```

〔화면 4〕 데이터 패턴 정의

```
                         Pattern File
                    Page  0  Clock Times
          1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16
DATA 1    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 2    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 3    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 4    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 5    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 6    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 7    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 8    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 9    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 10   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 11   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 12   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 13   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 14   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 15   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 16   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 17   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
DATA 18   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
UP  DOWN  RIGHT  LEFT  JUMP  EDIT  PRINT  *RANDOM  BINARY  SET  QUIT
```

같이 해주기 바란다.

```
↵
13 ↵                ↵
↵                   Y ↵
↵
↵
↵
↵
Y ↵
↵
↵
Y ↵
```

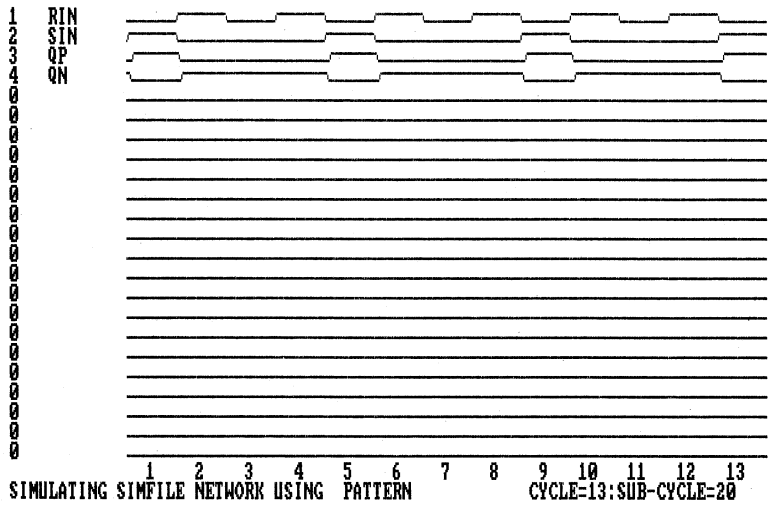
화면 5에 결과가 타이밍 차트 형식으로 나타났다. 결과가 맞는지 확인해 보자. RIN과 SIN는 리세트와 세트입력을 의미하며 QP와 QN은 Q의 양극과 음극을 의미한다.

클럭사이클 1, 2, 3에서 R=0, S=1이므로 QP = 1, R=1, S=0이므로 QP = 0, R=0, S=0이므로 데이터가 보존되어 OP = 0이 된다. 이후의 결과를 '디지탈강좌'에서 나타난 진리표의 값과 비교해 보면 서로 일치함을 알 수 있다.

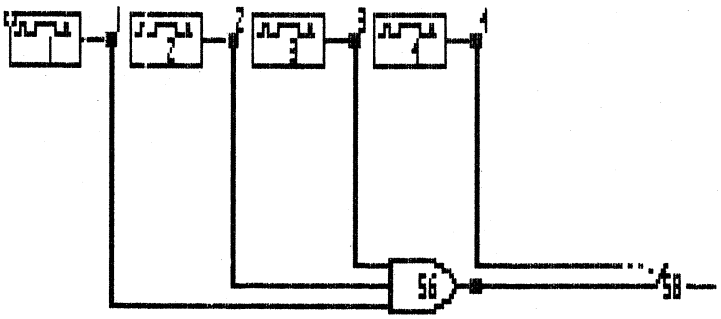
• 예제 2

화면 2를 보면 게이트 #15에 TGATE라는 것이 있다. 독자들은 이것에 대해 궁금하게 생각했을텐데 예제 2에서는

〔화면 5〕 타이밍도로 나타낸 시뮬레이션 결과



〔그림 3〕 게이트와 게이트의 연결



이것을 다루어 보도록 하겠다. 그럼 우선 논리회로를 그린 후 구체적인 설명을 덧붙이도록 하겠다. 그림 3의 회로를 그려보자.

디스켓을 초기화시키고 페이지 0을 선택한 후 다음과 같이 한다.

```
A
DATA 1 ↵     ┐
A ↵          ├ DATA 1을 그린다.
K            ┘

A
DATA 2 ↵     ┐
A ↵          ├ DATA 2를 그린다.
K            ┘

A
DATA 3 ↵     ┐
C ↵          ├ DATA 3를 그린다.
K            ┘

A
DATA 4 ↵     ┐
ENABLE ↵     │ 인에이블 신호
M            ├ DATA 4를 그린다.
M            ┘

A
AND3 ↵       ┐ 3입력 AND게이트를
CHIP1 ↵      ├ 그린다.
K            ┘

A
TGATE ↵      ┐
SELECT ↵     ├ T게이트를 그린다.
```

여기까지 6개의 회로 요소는 모두 그렸다. 게이트와 게이트 사이의 연결은 그림 2와 똑같이 C키와 커서이동키를 이용하여 연결해보기 바란다. 모두 연결이 되었으면 이 논리회로가 어떤 기능을 하는지 해석해 보기로 하자. Q를 눌러 저장시킨 후 디스크 메뉴에서 2를 눌러 분석모듈을 선택한 후 다음과 같이 입력한다.

```
2 ── 디스크 작업을 선택
3 ── 회로도 복원
↵ ── 리턴을 입력
```

위의 선택은 예제 1과 같다. 2와 3을 선택한 후 그냥 리턴키를 누른 것은 프로그램상에서 회로해석을 자동적으로 해주기 때문이다. 후에 회로 에디터에 대해 자세히 설명하겠다.

이제는 우리가 지정한 4개의 데이터에 대해 패턴을 정의해 보기로 하자. 역시 독자들이 화면 6과 같이 패턴 에디터를 수정하기 바란다. 다 되었으면 빠져나와 3을 선택해서 시뮬레이션에 들어가도록 하자. 옵션은 다음과 같이 하도록 한다.

```
↵
13 ↵ 13클럭까지 시뮬레이션
↵
↵
↵
↵
Y ↵
Y ↵
↵
↵
Y ↵
↵
Y ↵
```

결과는 화면 7과 같이 나타났을 것이다. 이 회로를 다음과 같이 해석할 수 있다.

ABC 3개의 입력에 의해 CHIP1의 AND3 게이트로 나온 출력을 T게이트에 연결했다. S타이밍도를 보면 인에

이블 신호가 1이면 CHIP1의 신호가 그대로 통과된다. 그러나 인에이블 신호가 0상태로 하강하면 하강하기 바로 전 상태(서브 사이클9)를 그대로 유지하게 된다.

•예제 3

전자회로에서나 논리회로에서나 가장 많이 쓰이고 강력한 기능을 가진 것이 플립플롭이다. 예제 3에서는 플립플롭 중에서도 가장 많이 쓰이는 JK-FF을 이용해 보겠다. 비동기식 16진 카운터를 구성하려고 하므로 독자들은 '디지탈강좌(4)'를 참조해 주기 바란다. 다음과 같이 입력하여 회로를 그려보자.

A
PHASE 1 ↵
CLOCK ↵
K
— 클럭을 그린다.

A
DATA 1 ↵
JIN ↵
K
— FF의 J단자 데이터를 그린다.

A
DATA 2 ↵
KIN ↵
M
M
J
— FF의 K단자 데이터를 그린다.

A
JKTTL ↵
QA ↵
K
— 토글에 의한 신호 QA를 위해 JK-FF를 그린다.

A
JKTTL ↵
QB ↵
K
— QB신호를 위해 JK-FF를 그린다.

A
JKTTL ↵
QC ↵
K
— QB신호를 위해 1/2 분주시킨 QC 신호를 위해 JK-FF을 그린다.

A
JKTTL ↵
QD ↵
K
— QC신호를 1/2 분주시킨 QD신호를 위한 JK-FF를 그린다.

〔화면 6〕 데이터 패턴 수정

Pattern File
Page 0 Clock Times

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DATA 1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0
DATA 2	0	1	1	0	1	1	0	1	1	0	1	1	1	0	0	0
DATA 3	0	0	1	0	0	1	0	0	1	0	0	1	1	0	0	0
DATA 4	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
DATA 5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DATA 6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DATA 7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DATA 8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DATA 9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DATA 10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DATA 11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DATA 12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DATA 13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DATA 14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DATA 15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DATA 16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DATA 17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DATA 18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

UP DOWN RIGHT LEFT JUMP EDIT PRINT *RANDOM BINARY SET QUIT

4개의 JK-FF과 2개의 데이터 그리고 1개의 클럭을 그렸다. 독자들은 **그림 4**를 보며 다음에 주의하여 회로를 완성해보기 바란다.

좌측 3개의 핀중 맨위의 것이 J입력이고 가운데 것이 클럭 입력이다. 그리고 맨 아랫 것이 K입력이다. 클럭은 QA 한곳에만 입력되고 J 입력 데이터는 각 FF에 모두 입력한다. K의 경우도 모두 입력된다. QB, QC의 경우 클럭은 앞단 FF의 2개 핀중 위의 것에서 연결된다. 우측 2개의 핀중 위의 것이 Q이고 아랫 것이 $\overline{Q}$이다.

선의 연결이 좀 복잡하니 천천히 해보기 바란다. 모두 되었으면 예제 1, 2의 경험을 살려 독자들의 취향(?)대로 시뮬레이션시켜 보기 바란다. 시뮬레이션 전에 꼭해야 할 것은 예제 1, 2에서와 같이 회로 복원 옵션에서 그냥 리턴을 치는 것과 또 하나는 패턴 에디터에서 데이터를 수정하는 것이다. 독자들은 DATA 1과 DATA 2를 모두 1로 가득 채워주기 바란다. 왜냐하면 '디지탈강좌'에서도 설명이 되었지만 JK-FF의 토글(J, K 모두 1일때) 특성을 이용하여 카운터를 만들기 때문이다. 회로가 제대로 그려졌고 바르게 패턴이 정의되었다면 올바른 시뮬레이션 결과를 얻을 것이다.

이제 회로가 바르게 시뮬레이션 되었나 살펴보자. 4비트로 된 16진 카운터인데 QD가 MSB(최상위 비트)이고 QA가 LSB(최하위 비트)가 되며 지금까지의 과정이 올바르게 되었다면 0001, 0010…으로 바르게 카운트 되는 것을 확인할 수 있을 것이다.

위의 3가지 예제를 통해 마이크로로직의 뛰어난 점 또는 불편한 점을 독자들은 느꼈을 것이다. 모든 프로그램에 장단점이 있으므로 독자들은 장점을 위주로 다음의 내용들을 계속 보아 주었으면 한다.

이번에는 분석모듈에 대한 설명에 들어가기 전에 회로를 그리는데 주의해야할 점 몇가지를 나열해 보겠다.

① 새로운 회로를 그리기 전에는 반드시 데이터 디스크를 초기화시켜야 한다.

② 하나의 핀과 출력 단자를 연결할 때는 출력 단자 바로 위에 연결해야 한다.

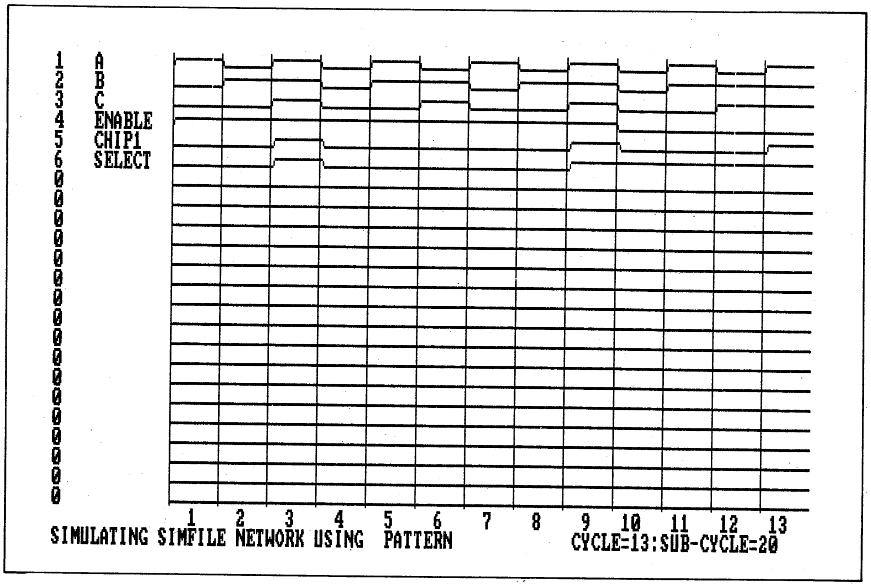
③ 접촉 심볼이 없는 곳은 연결된 것이 아니다.

(6) 분석 실행모듈

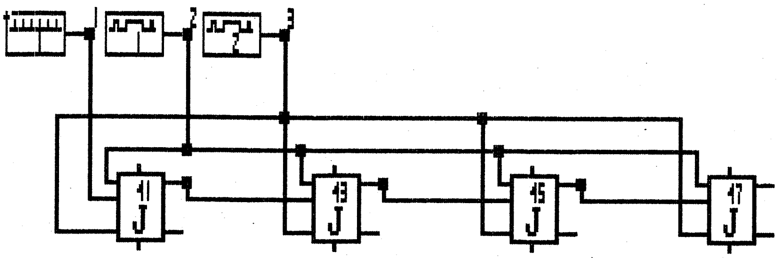
•회로 에디터

시뮬레이션이 끝난 후 이 옵션을 선택하면 구성된 소자의 종류와 연결상

〔화면 7〕 시뮬레이션 결과



〔그림 4〕



태가 나타나게 된다. 주된 기능은 회로 전체의 내용을 보여주거나 수정할 수 있게 하며 앞에서 자동으로 회로 에디터가 만들어진다고 설명한 바 있다. 이 기능이 왜 필요한가 의문이 생기게 되는데 이 회로 에디터는 매우 강력한 기능을 가지고 있다.

먼저 화면3의 설명을 한 후, 예제를 통해 그 기능을 알아보자.

① NO : 사용자 또는 프로그램이 부여한 게이트의 번호이다.

② NAME : 독자들이 회로를 그릴때 사용했던 게이트 고유의 이름이다. AND3, JKTTL… 등

③ LABEL : 회로의 기능을 잘 알아볼 수 있도록 사용자가 선택하는 이름이다. 6자 이내에서 사용할 수 있다.

④ A, B, … H : 해당 게이트의 핀 1, 2, …7의 이름, 즉 A=핀1이다. A, B, … H에는 숫자를 입력하는데 예를 들어 A밑에 4라고 쓰여 있으면 그 게이트의 1번 핀의 입력은 게이트 4번에서 들어 온다는 것이다.

• 디스크 작업

이 옵션을 선택하면 6가지의 기능을 얻을 수 있는데 화면에 나타난 글자 그대로의 기능, 에디터로 만든 회로를 저장하는 기능과 다시 불러 들일 수 있는 기능을 가진다.

• 시뮬레이션

이 옵션을 선택하면 다음과 같은 짧은 내용이 나타났다가 사라진다.

```
COMPILING NETWORK...PLEASE WAIT.
INITIAL NETWORK SIZE = xxx GATE
```

여기서 xxx부분에는 사용한 게이트의 갯수가 들어가게 된다.

다음은 여태껏 ↵ 으로 일관해 온 프롤로그 옵션에 대한 설명이다.

①SAVE(S), RECALL(R) OR NEW SIMULATION(N)

독자들은 가능하면 디폴트값인 'N'을 선택(그냥 ↵)하도록 한다. 'S'를 선택하게 되면 타이밍도의 첫번째 14 스텝을 저장하고 다시 시뮬레이션을 하게 된다.('R'은 이 타이밍도를 다시 볼 수 있도록 한다.)

②ENTER FINAL CLOCK CYCLE TO RUN

몇번째 클럭까지 시뮬레이션할까를 결정해 준다. 디폴트값이 256인데 한 화면이 보통 13이므로 예제에서는 13으로 바꾸었다. 적당한 값을 조절해 사용하기 바란다.

③ENTER THE NAME OF THE DATA CHANNEL PATTERN TO USE

디폴트는 현재 메모리속에 들어있는 패턴이 설정되므로 이 옵션으로 바꾸지 않고 사용하도록 한다.

④ENTER THE NO OF SUB-CYCLES PER CLOCK

한 클럭당의 서브 사이클수를 결정해 준다.

⑤⑥⑦ 옵션

이 세가지 옵션은 독자들이 값을 바꾸며 시뮬레이션 해보기 바란다. 정밀한 해석이나, 리포트 작성시 보기좋은 결과를 얻을 수 있을 것이다.

⑧PAUSE EACH SCREEN

이 옵션은 한화면이 웨이브 폼으로 가득 채워지면 일단 멈추게 하는 기능으로 항상 'Y'로 디폴트값을 바꾸어 사용하면 편리하다.

⑨USE CALCULATED(C), FIXED(F), PRIOR (P) OR UNIT(U) GATE DELAYS

이 옵션을 선택하면 게이트의 딜레이를 할당할 수 있게 된다. 'C'를 선택하면 모든 게이트의 팬아웃이 계산되

는데 다음공식에 따른다.

$$T=F+V*N$$

T : total delay
F : fixed delay
V : variable delay
N : fan-out

⑩SAVE RESULTS TO PRINTER(P), DISK (D) OR NEITHER(N)

디폴트값은 N으로 되어 있다. 필요에 따라 적당한 옵션으로 선택한다.

⑪COMPACT THE NETWORK

이 옵션은 게이트와 연결하지 않은 핀을 무시하거나, 빈 게이트를 제거함으로써 회로의 크기를 줄여준다. 그래서 이 옵션은 시뮬레이션을 줄여주고 회로를 해석하기 편리하게 만들어 준다.

⑫EDIT/REVIEW INITIAL CONDITIONS

이 옵션은 각 게이트의 초기상태를 정의할 수 있게 한다. 디폴트값은 '0'이다.

●유틸리티

유틸리티에는 4가지 기능이 있는데 이중 중요한 1, 2, 3번만 설명하도록 하겠다.

①ASSIGN NODES TO BE MONITORED

회로를 시뮬레이션할 때 23개의 출력을 한꺼번에 볼 수 있다(Apple의 경우 18개). 이 옵션을 선택함으로써 23개가 넘는 노드의 출력을 관찰할 수 있다.

②CREATING FAN-OUT REPORT

이 옵션은 구성한 회로가 제대로 되었나 확인할 수 있도록 게이트와 게이트의 연결관계를 모니터 화면이나 프린터로 출력해 준다.

③COMPARING SIMULATION FILES

이 옵션은 두개의 시뮬레이션된 화일을 비교해 준다. 예를 들어 Network1은 원래의 복잡한 회로이고 Newtwork2는 간략화시킨 것이라면, Network1과 Network2를 비교함으로써 간략화가 바르게 되었나를 알 수 있다.

●테스트 패턴 에디터

화면 4에 나타난 것처럼 사용자가 사용하는 데이터의 값을 조정할 수 있으며 수정방법은 화면 하단에 나와 있는데 다음예를 가지고 'B'의 기능을 살펴보기 바란다.

〔화면 8〕

```
                                    INPUT STATES
        PIN 1 0101010101010101010101010101010101010101010101010101010101010101
        PIN 2 0011001100110011001100110011001100110011001100110011001100110011
        PIN 3 0000111100001111000011110000111100001111000011110000111100001111
        PIN 4 0000000011111111000000001111111100000000111111110000000011111111
        PIN 5 0000000000000000111111111111111100000000000000001111111111111111
        LAST..0000000000000000000000000000000011111111111111111111111111111111

NO. NAME                 OUTPUT STATE TRUTH TABLE FOR ABOVE INPUT STATE

 1 INV          1010101010101010101010101010101010101010101010101010101010101010
 2 NOR3         1000000010000000100000001000000010000000100000001000000001000000
 3 NOR5         1000000000000000000000000000000010000000000000000000000000000000
 4 NAND3        1111111011111110111111101111111011111110111111101111111011111110
 5 NAND5        1111111111111111111111111111111011111111111111111111111111111110
 6 EX-OR        0110011001100110011001100110011001100110011001100110011001100110
 7 OR3          0111111101111111011111110111111101111111011111110111111101111111
 8 OR5          0111111111111111111111111111111101111111111111111111111111111111
 9 AND3         0000000100000001000000010000000100000001000000010000000100000001
10 AND5         0000000000000000000000000000000100000000000000000000000000000001
11 DFFTTL       0101010111111111000000000101010101010101111111110000000001010101
12 TFFTTL       0101010111111111000000000101010110101010111111111000000000101010
COMMAND MENU: U=UP D=DOWN E=EDIT C=COPY P=PRINT Q=QUIT
```

```
B ←── 바이너리 선택
1 ↵
2 ↵
```

●게이트 에디터

이 옵션에는 두가지 기능이 있는데 이것을 어떻게 조정하느냐에 따라 게이트에 편리하고 새로운 기능을 부여할 수 있다. 반면 잘못 다루면 예상치 못한 결과가 나오므로 기존에 정의되어 있는 기본게이트의 값은 바꾸지 않는게 좋다.

①TRUTH TABLE

이 옵션을 선택하면 **화면 8**을 볼 수 있는데 각 핀의 값에 따른 게이트의 결과가 맞는지 확인해 보기 바란다. D를 눌러 화면 8에 나타나지 않는 다른 게이트의 값도 살펴보기 바란다.

②GATE CHARACTERISTICS TABLE

화면 9를 보자. 이것이 특성 테이블(Characteristic Table)인데 좀 알아보기 힘든 모양으로 되어있다. 그 내용은 다음과 같다.

●Delay : Fixed(FIX)와 Variale (VAR)이 있는데 둘다 한 서브사이클당 지연(Delay)양을 나타낸다.

●Default Clocksin : 값이 0이면 디폴트의 신호가 사용된다.

●Default Clocksout : 값이 0이면 디폴트 핀번호의 신호에 의해서 출력전송이 생긴다.

●Clock Sampling Method for Pins 1, 2, 3, 4, 5 :

이 필드의 값이 1이면 포지티브 에지에서의 샘플, 0이면 모든 서브사이클 타임에서의 샘플, 2이면 네가티브에지에서의 샘플이다.

●클럭 웨이브 폼 에디터

다른 옵션과 마찬가지로 메뉴의 수정법은 같다. 입력된 데이터와 클럭의 실제모양은 아래 설명과 그림 5를 참고해주기 바란다.

Phase 2를 선택한다.

```
E
3 ↵
0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 ⌒
                          ⌒0 0 0 0 ↵
```

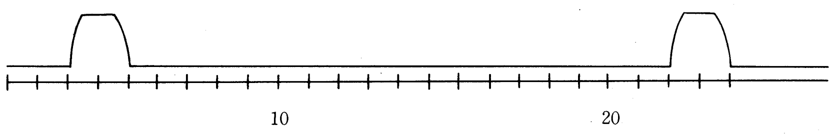
위의 데이터에 대한 클럭의 모양은 시뮬레이션 옵션 4에서 20(디폴트 값)이라고 했으면 그림 5와 같다. 그리고 첫번째 클럭(phase 0)은 정의되어 있

〔화면 9〕 특성표

NO.	NAME	DELAY FIX	DELAY VAR	DEFAULT CLOCKS IN	DEFAULT CLOCKS OUT	OR PIN #	CLOCK SAMPLING METHOD FOR PINS 1	2	3	4	5	OUT
1	INV	1	1	0	0	0	0	0	0	0	0	0
2	NOR3	1	1	0	0	0	0	0	0	0	0	0
3	NOR5	1	1	0	0	0	0	0	0	0	0	0
4	NAND3	1	1	0	0	0	0	0	0	0	0	0
5	NAND5	1	1	0	0	0	0	0	0	0	0	0
6	EX-OR	1	1	0	0	0	0	0	0	0	0	0
7	OR3	1	1	0	0	0	0	0	0	0	0	0
8	OR5	1	1	0	0	0	0	0	0	0	0	0
9	AND3	1	1	0	0	0	0	0	0	0	0	0
10	AND5	1	1	0	0	0	0	0	0	0	0	0
11	DFFTTL	1	1	0	0	2	2	0	2	2	2	2
12	TFFTTL	1	1	0	0	2	2	0	2	2	2	2
13	JKTTL	1	1	0	0	2	2	0	2	0	0	2
14	BUFFER	1	1	0	0	0	0	0	0	0	0	0
15	TGATE	1	1	0	0	0	0	0	0	0	0	0
16	DFFMOS	1	1	1	2	0	2	2	2	2	2	1
17	TFFMOS	1	1	1	2	0	2	2	2	2	2	1
18	JKMOS	1	1	1	2	0	2	2	2	2	2	1

COMMAND MENU: U=UP D=DOWN E=EDIT C=COPY P=PRINT Q=QUIT

〔그림 5〕



으므로 클럭을 재정의할 수 없고 1~9번만 정의해서 사용할 수 있다.

(7) 마이크로 로직의 첨가기능 활용

• 분할된 화면과 그 연결법

프로그램을 처음 시작할 때 페이지 0을 선택하여 사용했다. 그러나 복잡한 논리회로에서는 좀 좁다는 것을 독자들도 알았을 것이다.

마이크로 캡(Ⅱ)에서 4개의 화면이 있다고 설명한 바 있다. 마이크로 로직 역시 여러개의 다중화면을 가지고 있고 그 구성은 애플에서는 0~3까지, IBM에서는 0~8까지의 화면 구성으로 되어 있다.

그러면 두 화면이 어떻게 연결되는지 간단히 알아보기로 하자. 페이지0에서 커서를 화면 맨 우측으로 이동시킨다. 그리고 독자들이 아무 게이트나 하나를 그린다. 커서를 다시 그 게이트 출력단자에 옮기고 C로 접속한 후 화면우측 끝까지 그린다. 여기서 화면상단의 메뉴를 잠시 유의해서 보도록 한다. 라인옆에 게이트 내부에 써있는 숫자와 같은 숫자가 써있을 것이다. 즉 이 라인은 그 게이트에서 나온 라인을 의미한다.

여기서 다시 S를 눌러보자. 다음 질문이 나올 것이다.

ENTER DESIRED PAGE?

이때 페이지1을 선택한다.

1 ↵

화면(페이지0)이 저장되고 커서가 페이지1의 좌측에 나타난다. 다시 라인옆의 숫자를 보기 바란다. 페이지0에서와 같을 것이다. 이것으로 페이지0과 1은 서로 연결된 것이다.

• 하드 카피

서문에서도 언급했듯이 마이크로 로직은 진하고 깨끗한 하드카피본을 얻을 수 있다. AUTOEXEC.BAT을 만들때 DOS에 있는 GRAPHICS.COM을 사용했는데 마이크로 로직내에 있는 P 옵션보다 훨씬 낫다. GRAPHICS를 띄우면 각 화면의 메뉴에 나와 있는 프린트옵션동작이 안되며 〈SHIFT〉-〈Prtsc〉키로 화면을 하드카피 해야 한다.

결문

먼저 마이크로 캡에 대해서 좀 이야기 해야겠다. 지난 3월기사 끝부분에 "필자에게 연락해 주시면 성실히 답해 드릴 것을 약속한다."라는 망언(?)때문에 좀 고생을 했다. 여러분께서 문의해 주셨는데 답장해 드리기 위해 무척 애를 먹었다. 어떤 독자는 필자가 미처 발견하지 못한 좋은 기능을 알려 주시기도 했는데 진심으로 고맙게 생각한다. 이번 기사에서도 "성실히 답해 드릴 것을 약속한다."라는 약속을 하지 않을 수 없겠다.

컴퓨터가 대량으로 보급되고 친숙해져 가고 있다. 이에 따라 전문가나 전공인이 아니더라도 상당한 실력을 갖춘 사람이 주변에 많아졌다. 컴퓨터 초보단계에서 막 벗어나(퍼스널 컴퓨터를 대강 이해할 정도) 무엇인가 할 수 있을것 같다는 사람들도 훨씬 많아진 것은 사실이다.

그런데 이 단계에 있는 사람들이 자기가 아는 어떤 내용에는 대단한 것처럼 자만하고 어렵거나 모르는 것이 나오면 필요없는 것인양 무시해 버리는 경향이 있다. 자기가 알고 있는 지식이 있으면 필요한 사람들에게 나누어 줄 수 있고, 모르는 것이 있으면 후배에게도 물어보고 배울 수 있는 것이다. 컴퓨터에 막 접했거나, 익숙해지는 단계인 '퍼스널 컴퓨터' 독자들은 이 'MIND virus'에 걸리지 않길 바란다. PC

KB-88 시리즈

백인수 / 갑일전자 연구소

점차 정보화 사회로 전환되면서 컴퓨터는 이제 상당히 대중화되었다. 이와 함께 컴퓨터의 보급도 날로 증가하여 여러 기업체에서는 저가격대의 신제품이 계속 발표되고 있으며, 특히 중·고등학교의 정규과목으로까지 채택된 컴퓨터의 교육으로 그 수요가 점차 늘어날 전망이다. 그러나 학생들 개개인이 컴퓨터를 소유하기에는 아직도 고가의 장비이기 때문에 앞으로 좀더 가격이 낮춰져야 할 것이다.

이러한 문제점을 해결하기 위해 벌서 몇몇 업체에서는 교육용 PC를 속속 발표하여 시판중에 있다. 이러한 교육용 PC중 여기에 소개하는 KB-88 시리즈는 개방구조의 개념을 가지고 있어 필요에 따라 시스템을 구성할 수 있다는 장점을 가지고 있다.

OAS 란?

대부분의 PC는 마더보드라는 형식의 메인보드가 있어 여기에 CPU를 비롯한 메모리, 패리패럴 콘트롤러가 내장되어 있고 같은 메인보드내에 I/O 인터페이스를 접속할 수 있는 슬롯을 갖추고 있다. 그러나 이러한 형식의 구성은 XT, AT, 386 등 기종에 따라 각각의 메인보드를 구성하고 있어야 하기 때문에 케이스 또한 상이하다. 따라서 XT를 갖고 있는 유저가 AT나 386의 기종으로 향상시키고자 할 경우 새로 시스템을 구입하여야 하기 때문에 이중 비용을 부담하는 일이 생기게 된다.

OAS(Open Architecture Solution)는 이러한 단점을 보완하여 손쉽고 경제적으로 XT에서 AT 또는 386까지 구성할 수 있도록 설계된 획기적인 개방구조를 지칭한 것이다. 이 형식에서는 마더보드라는 개념을 탈피하여 백플레인(Backplane)이라는 용어를 사용하는데, 이는 마더보드에 있는 I/O 접속용 슬롯만을 따로 뽑아내어 구성한 것이다.

OAS시스템에서는 다른 PC에 있는 마더보드부분을 표준형카드로 구성하였으며 백플레인이라고 하는 슬롯에 카드만 삽입하면 그것이 바로 마더보드를 형성하게 되는 것이다. 따라서 삽입되는 카드의 종류에 따라 XT, AT 및 386 기종으로 구분되며, 한대의 기종으로 여러가지 기능을 수행할 수 있게 된다. 예를 들어 허클리스 비디오 카드를 사용하다가 CGA 비디오 카드가 필요하면 비디오 카드만 바꾸어주면 되듯이 이 시스템은 PC의 가장 중요한 요소인 마더보드까지도 간단하게 변경하여 필요한 업무에 맞게 사용할 수 있는 새로운 개념의 구성방식인 것이다. 그림 1은 기존의 마더보드와 이 시스템에서 사용한 OAS 구성을 보여준 것이다. 갑일전자에서는 이러한 OAS 체계에 맞는 XT용 메인보드와 286 AT용, 386SX용, 386용 메인보드를 갖추어 시판하고 있는 것이다.

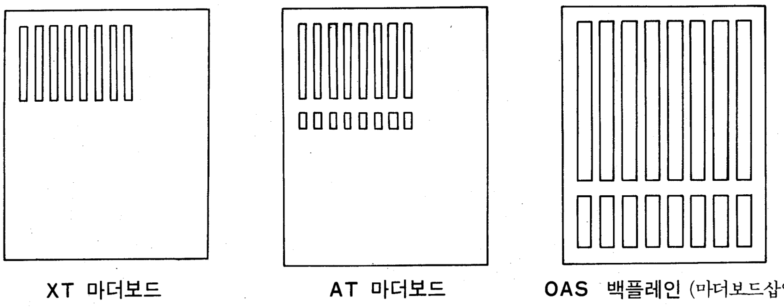
다음은 PC XT에서 대표적인 KB-88 시리즈의 구성과 메인보드의 기능에 대해서 설명하였다. 그림 2에서 시스템 내부의 구조를 보이고, 사진 1에 KB-88+의 조립방법을 보여주고 있다.

종류

KB-88 시리즈에는 교육용인 KB-88E와 업무용인 KB-88 및 KB-88+ 등 3가지 종류가 있다. 이 시스템은 모두 OAS의 개념으로 구성, 설계되어 있으며, KB-88E와 KB-88에서 백플레인에 대한 슬롯수는 4개이고, KB-88+는 8개의 슬롯을 가지고 있다.

여기에 대한 자세한 사항은 표 1에 비교해 놓았다.

〔그림 1〕 마더보드의 비교



특징

KB-88 시리즈에 사용되는 메인보드가 표준형 카드로 되어 있다고 앞에서 언급하였다. 이것은 기존의 메인보드에서 가지는 어떤 기능보다도 많은 기능을 가지고 있는데, 우선 자체내에 FDD 제어회로, 직렬포트 2개, 병렬포트 1개, 메모리 제어회로, 클럭/칼렌더 기능등이 내장되어 있으며 더욱 특징적인 것은 하드디스크 제어회로가 내장되어 있다는 것이다.

일반적으로 플로피디스크는 물론, 하드디스크 제어를 위하여 별도의 인터페이스 카드가 필요하지만, KB-88 시리즈의 메인보드는 인텔리전트(Intelligent) 하드디스크 제어회로가 들어 있는 코넥터만 연결하면 사용할 수 있다.

KB-88시리즈 메인보드의 구성

KB-88 시리즈의 메인보드는 다음과 같이 구성되어 있고, 그림 3은 이 회로들의 배치를 보여주고 있다.

① CPU 회로
② 메모리 제어회로
③ FDD 제어회로
④ HDD 제어회로
⑤ 주변기기 접속회로(I/O)
⑥ 기타

(1) CPU회로

표준형 카드내에 많은 기능을 갖도록 하기 위해 KB-88 보드는 Intel의 와일드카드(Wildcard)라는 부품을 사용하였다. 와일드카드는 8088 CPU를 비롯하여 DMA 제어회로, BUS 제어회로 등 많은 부분을 집적한 카드이다(그림 4 참조). CPU의 동작속도는 처음 IBM PC/XT가 4.77MHz로 동작이 되도록 설계되었고, 점차 빠른 CPU가 나와 4.77, 7.15, 9.54MHz 등이 있으며, 일반적으로 5, 8, 10MHz로 불리워지고 있다. KB-88 시리즈는 7.15MHz로 동작되도록 프로그램되어 있으나, I/O PORT 63H에 특정한 값을 주는 프로그램에 의하여 다른 속도로 변환할 수 있다.

CPU에 공급되는 클럭주파수는 14.31818MHz이나 10MHz 이상 수행하기 위해서는 28.636MHz(본 시리즈에 채택)의 클럭이 필요하다. 참고로 클럭의 공급에서 I/O 슬롯의 B30번 핀(OSC)은 CPU에 입력되는 주파수와 같고, B20번 핀(CLK)에는 CPU가 동작되는 클럭이 공급된다. 또한 CPU회로에 사용하고자하는 비디오카드의 종류를 선택하는 점퍼스위치가 나와 있으며, 연산용 프로세서인 8087(287)을 삽입할 수 있는 소켓과 하드웨어 초기화를 위한 리세트스위치가 핀으로 나와있다.

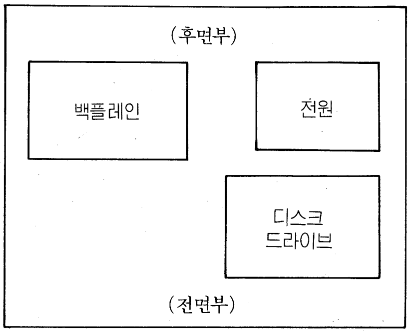
다음은 점퍼스위치에 의한 비디오모드 선택방법이다.

J1	J2	
0	0	: 80×25 허큘리스
0	1	: 80×25 컬러그래픽(CGA, EGA, VGA 자동선택)
1	0	: 40×25 텍스트 컬러
1	1	: 사용안함

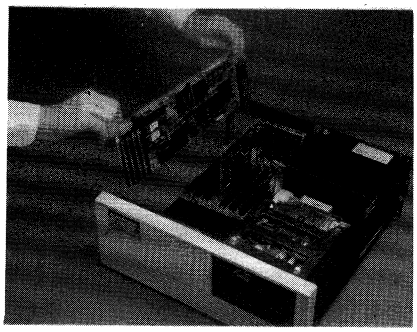
(2) 메모리 제어회로

메모리의 구성 방법은 각사에 따라 256K×1비트 DRAM 혹은 256K×4비트 DRAM 등 여러가지가 있을 수 있으나 KB-88은 256K×1 BIT DRAM

〔그림 2〕 시스템 내부 구조



〔사진 1〕 KB-88+조립방법



〔표 1〕 KB-88 시리즈 비교

구 분	KB-88E	KB-88	KB-88+
용 도	교 육 용	업 무 용	
CPU	8088 (286, 386 선택)		
클 럭	4.77/7.15/9.54MHz		
RAM	256 KB (MAX 640KB)	512KB (MAX 640KB)	
FDD	360KB× 1 대	360KB× 2 대	
HDD	20 또는 40MB (선택)		
I/O	시리얼포트 2개, 패러럴포트 1개		
키보드	86키		
슬 롯	4 개		8 개
파 워	116 W		150 W
외 형	114×386×395 (HWD)		155×406×414
기 타	리얼타임클럭 내장, 키 잠금 장치 286이상 RAM은 SIMM 메모리 (2MB) 확장지원 FDD, HDD 콘트롤러 내장		

을 사용하였다. 이 DRAM의 속도는 80ns, 100ns, 120ns, 150ns 등이 있는데 이 시스템에서는 100ns의 DRAM을 사용하여 실행속도의 보조를 맞추어 주고 있다.

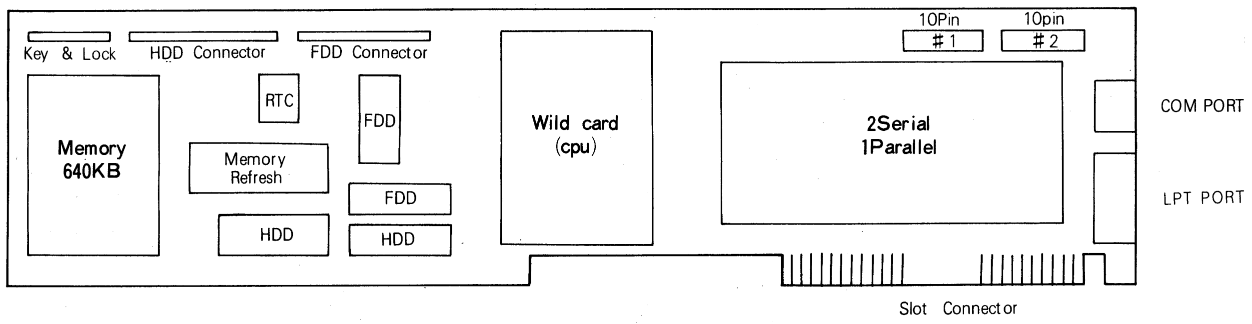
램 소켓의 8개 세줄중 한줄을 삽입하면 256K비트×8개=256K 바이트의 메모리 구성이 되어 기본적인 시스템으로 동작시킬 수 있으며, 512K의 메모리를 필요로 할 경우 16개, 640K는 24개의 DRAM을 이용하여 구성할 수 있다.

또한 메모리 에러 검출을 위하여 체크섬비트(checksum bit)용으로 1개씩 더 사용하는 경우(256KB 구성시 데이터 8개, 체크섬 1개)도 있으나 일반적으로 XT에서는 체크섬비트를 사용하지 않는다. 메모리확장시 보드에 소켓이 준비되어 있어 XT에서는 640K, AT에서는 SIMM(Single In Memory Module) 메모리(소켓 삽입용)를 사용하여 최대 4MB까지 확장할 수 있으나 현재 2MB SIMM 메모리만이 지원되고 있다. 메모리는 초기화시에 시스템을 운영하기 위한 각종 정보(I/O 상태, 변수 등)를 저장하며 사용자가 작성한 프로그램의 운영에 사용되고 인터럽트 서비스를 위한 어드레스 등이 들어있다.

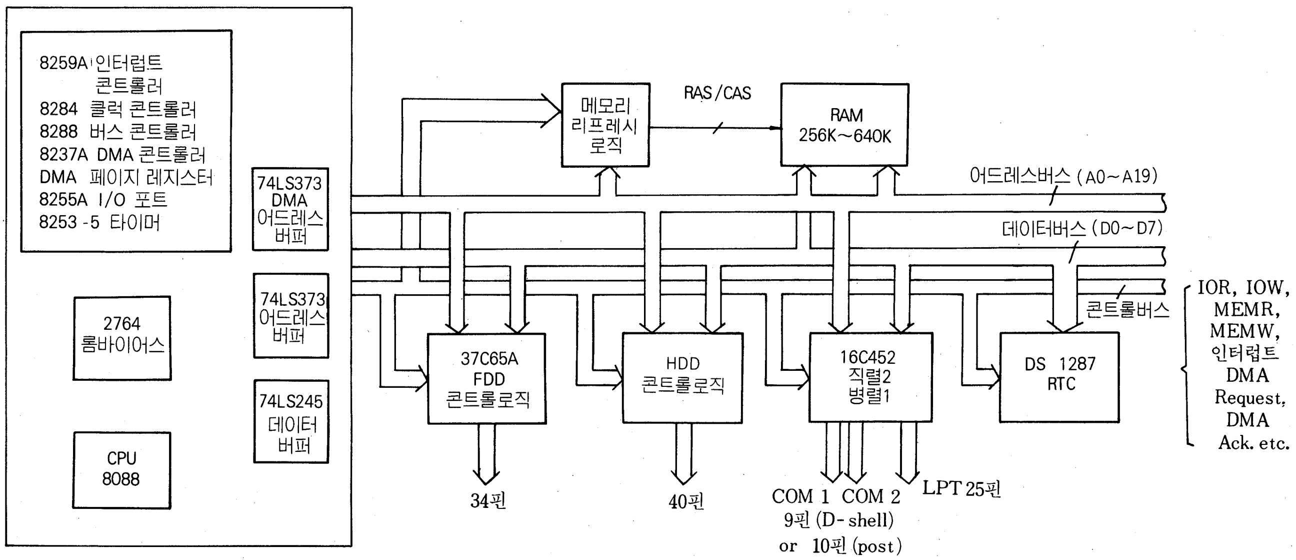
(3) 플로피디스크 제어회로

대부분 하나의 시스템에 두개의 FDD를 부착하는 것이 보통이다. 연결시 34핀 코넥터를 사용하는데 FDD 콘트롤 포트에서 첫번째 코넥터가 드라이브 B용이고, 라인 일부가 꼬여져 있는 것이 드라이브 A에 연결된다. FDD 연결시 주의할 점은 FDD내에 자체적으로 동작 드라이브를 설정하는 점퍼를 핀(0~4 등)으로 설정할 수 있는데 드라이브 A인 경우는 "0"으로, B인 경우는 "1"로 점퍼를 연결하여야 한다. 그

〔그림 3〕 메인보드의 구성



〔그림 4〕 와일드카드 구성도



(8087은 옵션)

렇지 않을 경우 드라이브의 에러가 발생한다. 또 하나 주의할 점은 점퍼옆에 터미네이터(Terminator)라는 저항이 삽입되어 있어야 한다.(소켓으로 되어 있는 경우도 있음) 이것 역시 없을 경우 데이터라이트시 데이터파손 등의 에러가 발생할 수 있다. FDD동작을 위한 4개의 전원선은 +12V, +5V와 2개의 GND로 구성되며, 이것은 5.25인치 드라이브의 동일한 규격이다.

3.5인치 드라이브가 최근 들어 많이 사용되고 있는데 이 드라이브를 연결할 경우는 별도의 콘트롤러를 슬롯에 삽입하여야 한다. 이 경우 인터페이스 코넥터와 전원 코넥터가 상이하므로 연결시 판매점에 문의한 후 사용하는 것이 안전하다.

대부분의 시스템에서 사용할 수 있는 드라이브의 수는 2대 이므로 그 이상을 접속하고자 할 경우 별도의 콘트롤러를 사용하여야 한다. 그리고 제어회로에서 사용하는 콘트롤러는 Western Digital의 37C65칩을 사용하고 있다.

(4) 하드디스크 제어회로

앞에서 언급하였던 것과 같이 KB-88 메인보드내에 별도의 하드디스크 제어용 카드가 없어도 20MB의 하드디스크를 제어할 수 있는 회로를 가지고 있다. 즉 하드디스크의 BIOS와 40핀 코넥터로써 손쉽게 20MB의 하드디스크를 장착할 수 있는 것이다.

이렇듯 메인보드 자체에서 제어되는 하드디스크는 1대로 제한되어 있으나 약간의 회로변경으로 2대의 접속도 가능하다. 그러나 이것은 회로상의 변경이므로 하드디스크 추가 사용시 별도의 콘트롤러를 삽입하여 사용하도록 권한다. 하드디스크는 초기화된 후에 드라이브 C, D, E 등으로 불리워지며 연결시 주의사항은 FDD와 같다.

메인보드상의 HDD 제어회로로 미니스크라이브와 웨스턴 디지탈사의 디스크를 사용할수 있으며, 별도의 콘트롤러를 사용할 경우 카드에 맞는 하드디스크를 사용할 수 있다. XT에서의 최대 사용용량은 20MB이다. 그러나 구입시 옵션으로 그 이상도 가능하다.

FDD를 포맷팅하기 위해서는 제일 먼저 디스크를 초기화하여야 하는데 DEBUG 소프트웨어를 이용하여 수행한다. DEBUG라는 명령어와 ENTER키를 치면 마이너스 프롬프트가 나오는데 여기에 G=C800:5를 입력한 후 ENTER키를 치면 디스크 초기화에 필요한 값들을 입력시키기 위한 질문사항이 나오고, 모두 세트되면 초기화를 시작한다. (수행방법은 콘트롤러 공급업체마다 다를 수도 있음) 초기화후 하드디스크의 동작영역을 규정하기 위한 "FDISK"와 섹터구분을 위한 디스크 포맷을 수행하여야 한다. 하드디스크에서 어떤 에러발생시 "1701"이라는 메시지가 나타나는데, 이것은 하드디스크 에러시 공통적으로 나타난다. 이럴 경우 전원 및 코넥터, 점퍼위치 등을 확인하고, 모두 정상인 상태에서도 에러가 발생하는 경우 하드디스크가 파손되었거나 그외 콘트롤러 및 시스템의 문제일 수도 있으므로 구입처에 문의하기 바란다. AT에서의 초기화는 "IDISK"라는 명령어가 있으며, 다음 단계는 위와 동일하다.

(5) 주변기기 접속회로

주변기기와 접속하여 통신하기 위해서 기본적으로 2개의 직렬포트와 1개의 병렬포트를 갖추고 있으며, 직렬포트는 RS-232C라는 방식을, 병렬포트는 센트로닉스라는 방식을 채택하고 있다. 직렬포트는 2개의 I/O 어드레스(I/O 2F8H, 3F8H)에서 동작되며, 9핀의 코넥터가 외부로 나와 있고, 내부에 10핀의 코넥터가 두개 있어 25핀 또는 기타의 코넥터로 사용하고자 할 경우에 이용된다. 병렬포트는 프린터와의 접속을 위해 사용되며 25핀 코넥터가 사용된다.

병렬포트를 이용하기 위해서는 I/O 어드레스 278H 혹은 378H를 제어하면 되는데 어드레스 선택은 점퍼스 위치로써 손쉽게 바꿀 수 있다.

직렬포트와 병렬포트는 각각 점퍼스 위치에 의하여 사용가능(Enable)과 불가능(Disable) 혹은 COM1/COM2 선택, LPT1/LPT2 선택, 인터럽트서비스 요구, 라인변경 등을 선택할 수 있으므로 기존의 다른 카드(모뎀, 비디오등)와 관계없이 사용 가능하다.

메인보드에 구성되어 있지 않은 비디오카드 등은 슬롯을 통하여 메인보드와 서로의 상태를 점검하는데, XT인 경우에는 8비트 슬롯(62핀)을, AT 이상인 경우는 16비트 슬롯(98핀)을 이용한다. 슬롯에 있는 각 핀의 동작은 IBM PC 슬롯의 동작과 동일하다.

각 옵션카드는 카드 자체의 어드레스에서 동작하도록 설계되어 있으므로

〔표 2〕 KB-88 I/O 어드레스 맵

어드레스	I/O
000-00F	DMA 콘트롤러
010-01F	예비
020-021	인터럽트 콘트롤러
022-03F	예비
040-043	타이머
044-05F	예비
060	키보드 데이터 레지스터
061	콘트롤 레지스터 0
062	콘트롤 레지스터 1
063	콘트롤 레지스터 2
064-06F	예비
070-07F	사용안함
081-083	DMA 페이지 레지스터
084-0BF	예비
0A0	NMI 마스크 레지스터
0A1-0FF	예비
278-27F	프린터(LPT2)
2F8-2FF	비동기 통신(COM2)
320-32F	Fixed Disk(옵션)
370-377	리얼타임클럭(옵션)
378-37F	프린터(LPT1)
3B0-3BF	모노크롬 디스플레이
3D0-3DF	컬러/그래픽
3F8-3F7	디스크
3F8-3FF	비동기 통신(COM1)

초기화시에 메인보드는 특정 어드레스를 통하여 각각의 슬롯을 확인하게 된다. 표 2는 입출력 어드레스 맵을 나타내고 있다. 예를 들어 비디오카드가 있는지 확인하기 위하여 I/O 어드레스 3D0~3DFH까지에 어떤 값을 주어 카드가 있을 경우 어떤 반응이 나타나고, 없을 경우 아무런 반응도 나타나지 않는다. 이러한 일련의 동작이 슬롯을 통하여 이루어지며 이상 유무를 점검한다. 그리고 기타 주변기기의 접속에서 슬롯 4개가 있는 시스템은 좀더 확장을 요구하는 사용자에게 약간의 제약을 줄 수 있지만 슬롯 8개 짜리는 충분히 연결하여 사용할 수 있을 것이다.

(6) 기타

클럭과 칼렌더의 기능을 갖는 리얼타임클럭 회로로써 Dallas 1287을 사용하고 있어 고도의 정확성과 영구성을 가지며 사용자로 하여금 문서 혹은 기타 프로그램의 작성일과 시간 등을 알 수 있게 하여 준다.

전원을 껐다가 켤 필요없이 손쉽게 하드웨어를 재기동시킬 수 있는 리세트버튼이 있으며, 사용자를 제한할 수 있는 키잠금장치와 전원공급상태 및 하드디스크의 사용중임을 알 수 있는 표시장치가 되어 있다.

비디오 카드는 허큘리스 카드가 기본적으로 사용되며 한글을 사용할 수 있도록 프로그램이 공급되고 있다. 또한 행정전산망용 및 조합형 한글 코드를 지원하기 위한 비디오 카드가 선택사양으로 있으며, 이들을 위한 HM-100, HM-200 모니터를 지원하고 있다.

OAS 특성상 XT용 메인보드 대신 AT용 메인보드를 사용할 경우 대부분의 비디오카드와 I/O카드를 그대로 사용하지만 FDD를 AT용으로 교환하여 주어야 사용상 문제점이 없을 것이다.

이상과 같이 KB-88 시리즈는 부피가 작지만 많은 기능을 가지고 있어 소형화, 집적화에 적합하다.

한 가지 이 시스템에서 제외된 것은 마우스 콘트롤러와 게임포트의 인터페이스이다.

보다 편리하고 우수한 성능, 그리고 저가의 컴퓨터 보급을 위한 방법의 하나로 오픈 아키텍처 솔루션(OAS)이라는 방식을 채택한 시스템이 제한된 크기에서도 다양한 기능을 가질 수 있어 앞으로 OAS 방식의 시스템이 많이 보급되리라 본다. PC

한글 데이터베이스 프로그램

엘렉슨에서는 한글처리와 사용방법의 용의성에 주안점을 두고 국내 업무구조에 적합하도록 한글 데이터베이스 프로그램인 오아시스-V를 개발, 발표했다. 이 프로그램은 각 모듈별로 인공지능 방식을 채택했으며 WP/LOTUS에 저장되어 있는 데이터와의 커뮤니케이션도 가능하도록 되어 있는 등 그 기능이 매우 다양하다. 여기에서는 이 오아시스-V의 특장과 기능에 대해 간략하게 알아본다.

초기 컴퓨터의 활용은 워드프로세서를 이용한 문서작성이나 스프레드시트를 이용한 간단한 계산을 하는 정도였으나 점차 사용자들은 좀더 많은 부분에서 컴퓨터를 활용하길 바랬다. 이러한 요구로 dBASE나 제이블린(Jevelin), Q&A, 패러독스(Paradox), 파워베이스(Power Base) 등과 같은 업무용 프로그램들이 수없이 등장했지만 그중에서도 dBASE가 가장 많이 활용되었고, 그외의 것은 사용이 불편한 점 등의 여러가지 문제로 거의 국내에서는 활용되지 않고 있는 상황이다.

그리고 현재 dBASE가 가장 많이 사랑을 받고 있는 이유는 그 기능자체가 강력하다는 점도 있지만 프로그램에 대한 지식이 없는 사람도 프로그램할 수 있도록 쉬운 방법을 채택했다는 점이다. 그러나 국내 업무에 적용하기까지는 여전히 그 프로그램 작성방법은 복잡하고, 최소한 MS-DOS는 익혀야 할 뿐아니라 한글처리가 제대로 되지 않아 프로그램하는데 필요한 각 명령어를 거의 마스트를 해야 하는 등의 불편한 점이 있다. 이로 인해 초보자들은 이 dBASE를 익히기에 앞서 두려움을 갖기 쉽상이었다.

이러한 상황에서 dBASE와 같은 맥락으로 한글처리와 사용방법의 용이성을 고려한 오아시스-V의 등장에 관심이 가는 것은 당연한 이치이다.이 오아시스-V는 지난 85년 엘렉슨이 자체 개발한 오아시스 한글 데이터베이스가 모체가 된다. 이 데이터베이스는 8비트 전용 프로그램이었기 때문에 1년만인 86년 11월에 다시 16비트로 전환한 오아스-M을 발표하게 되었다. 이 프로그램은 베이직과 어셈블리로 작성되었고 이의 비율은 베이직의 거의 95%였고 어셈블리어가 5%였다.

그러나 그 뒤 또 다시 유저들의 요구사항을 수렴하여 업그레이드, 오아시스-V를 발표하게 된 것이다. 이 프로그램의 경우는 어셈블리어의 비중이 70%로 거의 중요한 제어부분을 콘트롤하도록 하였고 베이직은 퀵베이직으로 단순한 제어에만 관여하도록 하였다. 그리고 각 프로그램 모듈별로 인공지능방식을 채택한 것이 특이하다.

이는 한국형 데이터베이스라고 할만큼 국내의 업무실정에 가장 적합한 통합형 소프트웨어로 평가되고 있다. 왜냐하면 오아시스-V 하나로 인사급여관리, 자재수불관리, 판매재고관리에서 무역관리, 일반회계관리에 이르기까지 약40~50여개의 단위업무를 처리할 수 있어 소수의 인원으로 초기 전산화를 추진하는 회사나 전산화에 실패한 중소기업, 그리고 표 1에 보인 바와 같이 그동안 외주용역에 의존하던 업체도 자체기술로 해결할 수 있게 하였기 때문이다.

그러나 특히 아직까지 국내실정은 컴퓨터마인드가 깊이 뿌리내리지 못한 상황이라 컴퓨터를 전문으로 취급하는 회사를 제외하고는 거의 컴퓨터에 대한 전문지식을 갖고 있는 실무자가 드물다. 따라서 이 프로그램은 그러한 실무자를 우선적으로 고려하여 일반적으로 우리가 생각할 수 있는 형태의 프로그램을 작성을 하지 않고 스크린상에서 한눈에 볼 수 있도록 메뉴방식을 채택하여 사용을 용이하게 했다. 뿐만 아니라 한글처리까지 했다는 점에 그 의미는 크다 하겠다.

오아시스-V의 특징 및 기능

오아시스-V는 앞서 말한 바와 같이 컴퓨터에 대한 전문기술이 없는 사람이라도 현재 맡고 있는 업무를 잘 파악하고만 있으면 그 형태대로 직접 프로그래밍할 수 있다. 이미 프로그램이 짜여진 상태일 경우에는 업무의 형태, 보고서 출력양식, 자료의 관리형식 등이 메뉴형식으로 화면에 나타나기 때문에 이를 단순히 지정하는 것만으로도 충분히 업무에 적용할 수 있도록한 프로그램으로 국내에서 개발된 IBM-PC용 범용 소프트웨어의 하나이다.

이 프로그램의 구성은 그림 1

〔표 1〕 용역 프로그램과 오아시스-V 프로그램의 비교

	용역 프로그램	오아시스-V
개발비용	업무당 100~300만원	없음
개발시간	업무당 2~3개월	업무당 48시간
프로그램 개발	용역회사	현업실무자
업무수정	시간/인력/비용 추가	즉시처리/비용없음

오아시스-V

고 같이 그림 1에서 볼 수 있는 구성들이 오아시스-V의 메인 메뉴를 이룬다. 이 메인 메뉴는 화면 1과 같다. 이 메인 메뉴에 각 기능을 수행할 때에는 사용자가 선택하여 사용하면 되는데 그 선택방법은 일반 메뉴방식과 별차이는 없다. 단지 이 메뉴를 거의 익혀 불필요함을 느낄 때는 삭제하고 사용자 자신이 짠 메뉴를 메인 메뉴로 만들 수 있다. 그리고 사용자가 화면 1의 메인 메뉴상에서 자체업무내용을 프로그램할 경우, '화면 편집'을 선택하면 프로그램할 수 있는 초기화면으로 빈화면이 나오면서 프로그램이 가능하게 된다.

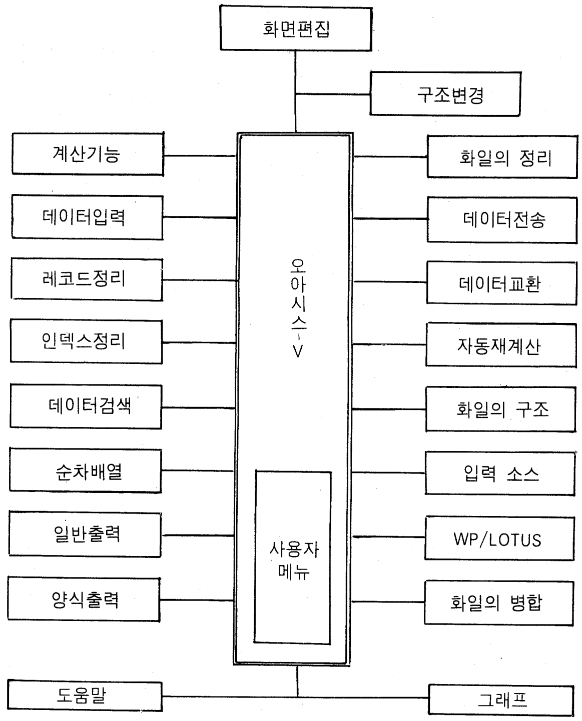
오아시스-V는 과연 통합형 프로그램이라할 만큼 그 기능이 매우 다양하다. 즉 다중입력이란 특수한 방법으로 데이터 입력시 시간과 노력을 절감할 수 있고 데이터 입력시에도 언제나 외부 화일의 검색을 할 수가 있어서 사용자는 반복되는 작업을 줄일 수가 있다. 또 화면검색을 이용하여 조건에 맞는 데이터를 일괄적으로 검색할 수가 있는데 검색조건에 AND나 OR를 비롯하여 다양한 조건을 줄 수가 있다. 또한 인덱스를 지정하면 마스터 화일인 경우에 자동으로 처리해주며 인덱스항목의 중복 데이터를 자동 검출해 준다. 그 외에도 이 프로그램이 가진 기능들은 다음과 같다.

① 검색된 데이터의 수정 입력
② 사용자가 정의하는 데이터 검색용 화면
③ 부분적인 순차 배열
④ 마스터화일의 데이터 자동 출력
⑤ 데이터 출력시 간이언어를 이용한 계산식 적용
⑥ 양식(FORM)형태의 출력
⑦ 임시화일을 이용한 화일의 자동 UPDATE
⑧ 데이터의 전송 및 교환
⑨ 특정자료의 일괄처리를 위한 자동 재계산
⑩ 사용된 화일들의 구조 및 계산식 정보 제공
⑪ 입력 소스프로그램 재생
⑫ 워드프로세서/로터스와 데이터 커뮤니케이션
⑬ 화일과 화일의 데이터 병합(Merge)
⑭ 사용자 메뉴의 작성 및 활용
⑮ 각 프로그램 기능에 대한 도움말(Help)

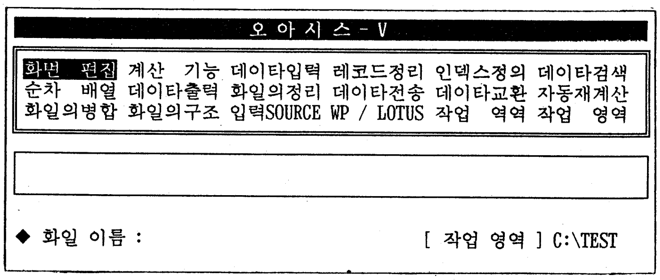
이와 같이 국내업무구조를 수렴한 한국형 업무용 프로그램이라는 점과 인공지능방식을 채택했다는 점 외에도 화면편집(File Design)이 워드프로세서를 사용하는 것처럼 용이하다는 점, 초보자도 사용하기 쉬운 간이언어를 사용한 점, 새로운 기법의 HISAM 및 메모리 콘트롤러와 V-SORT를 개발한 점, 사용자가 메뉴를 직접 작성하여 종합적인 업무처리를 할 수 있다는 점들이 오아시스-V의 주요 특징이다.

또한, 오아시스-V는 IBM-PC/XT 또는 AT와 호환성이 있는 16비트 이상의 컴퓨터에서

〔그림 1〕 오아시스-V 프로그램의 구성



〔화면 1〕 메인메뉴



사용할 수 있으며 640KB 메모리와 10MB 이상의 하드디스크를 구비하여야 사용할 수 있으나 현재로는 컴퓨터의 가격이 점차 저렴해지고 있다.

그러나 삼보한글카드가 반드시 장착되어야 한다는 점이 단점이다. 따라서 이를 구비하지 못한 유저들에게는 조금 아쉬운 감이 있다.

그러면 지금까지 설명한 오아시스-V의 특수한 기능과 일반적인 기능에 대해서 알아보자.

(1) 특수한 기능

•화면편집과 계산 기능

이들 특성에 대해 좀 덧붙힌

다면 화면편집의 경우 화면문자체의 속성변환이나 그래픽을 이용한 드로잉(Drawing) 그래픽 모드로 다양한 화면처리, 화면 전체의 스크롤 등이 간단히 처되므로 없었는데 확대출력(Doub-

이러한 특성들을 이용하여 사용자는 필요한 항목을 나열하는 형식으로 화면 2와 같이 짜면 된다. 화면 2는 사용자가 업무에 필요한 항목의 이름과 길이를 지정하여 "제품수불부"란 하나의 화일이 작성된 예이다. 회사나 업무에 따라서 항목의 추가나 삭제 또는 구조변경 기능을 이용한 새로운 "제품수불부"를 만들 수 있으므로 사용자 임의대로 사용이 가능하다. 그리고 1화일에 최대 200항목과 최대 5페이지의 화면이 사용가능하다 (표 2 참조).

간이언어는 일반적으로 어렵고 복잡하게만 느껴지는 프로그램언어 및 논리를 서술하는 방법을 사용하지 않고 간단하게 처리할 수 있도록한 언어로서 dBASE에서도 dBASE 간이언어를 이용하여 업무처리를 하고 있지만 화일간의 연결 등 완벽한 업무를 처리하려면 하나의 로직이 형성되어야만 한다. 따라서 오아시스-V에서는 다양한 계산식(함수나 조건문)은 물론 간단한 지정만으로 화일간의 연산까지 처리되도록 간이언어를 로직화하였다.

예를 들어 실제 업무에서 '공급가=수량✱단가'를 계산한다고 했을 때 이를 오아시스-V에 적용해 보면 '!12=!10✱!11'로 그 계산이 이루어진다. 여기서 !은 사용하는 화일의 항목을 지정하는 문자이다. 그리고 '출고가=공급가+부가세-할인액'에 대한 계산을 할 경우 '!15=!12+!13-!14'로 이루어진다.

〔화면 2〕 화면편집 예

```
                         제 품 수 불 부

일    자: ████         구    분: █ [1.입고 2.출고 3.반품]
        (1)                     (2)

회사코드: ███           회사명: ██████          담당자: ██████
        (3)                    (4)                    (5)

제품코드: ███           품    명: ██████         재    고: ██████
        (6)                    (7)                    (8)

규    격: ██████       단    가: ██████         수    량: ██████
        (9)                    (10)                   (11)

공급가: ████████       부가세: ████████         할인액: ████████
        (12)                   (13)                   (14)

출고가: ████████       수금액: ████████         잔    액: ████████
        (15)                   (16)                   (17)
```

〔표 2〕 오아시스-V 프로그램의 규격

화일의 항목수	200항목
화일의 화면 수	5페이지
화일당 레코드 수	999,999 레코드
레코드의 문자 수	8,000바이트
인덱스 항목 수	3항목
유효자리수	16자리
최대표현수	+1.8E+63
최소표현수	-1.8E-63
마스터화일	8화일
목록(TABLE)화일	3화일

특히 IF...THEN...문을 사용하여 '만약 구분이 2이면 공급가=수량✱단가를 취하고, 아니면 0을 취한다'라는 내용을 표현할 경우에는 'IF !2=2 THEN !12=!10✱!11 ELSE !12=0'과 같이 할 수 있다. 이때 소숫점은 2자리까지 자동 계산된다.

• 데이터 입력

데이터 입력에는 사용자가 편집한 화면에 표시된 항목을 그대로 이용하여 1레코드씩 입력하는 단일 입력방법과 화면에 표시된 항목을 바꾸어 여러 레코드를 동시에 입력하는 다중입력방법 2가지가 있다.

특히 다중입력이 가능하기 때문에 전표와 같이 데이터의 입력값이 항상 변하는 경우에는 다중입력영역을 설정하여 쉽게 할 수 있어 시간과 노력이 절감된다.

또한 데이터 입력시에도 언제나 외부화일을 검색할 수가 있어서 사용자는 반복되는 작업을 줄일 수 있다. 그리고 한글을 입력시 설정영역보다 넘었을 경우 발생될 수 있는 한글파괴문제는 스스로가 보완하고 있다. 한글 1자는 일반 워드프로세서와 같이 2바이트(영문 2자)이다. 문자체는 N(Normal), U(Underline), R(Reverse), 하이라이트(Highlight), 하이라이트 언더라인(Highlight Underline) 등 5가지 형태로 표현할 수 있다.

• 데이터 검색

화일에 입력된 데이터를 검색하는 것으로 순차적인 검색과 인덱스 검색방법이 있어 각각 따로 사용할 수도 있지만 병행 사용이 가능하다. 그리고 문자와 숫자로 구분하여 다양한 검색조건을 부여할 수 있는데 여기서 각 항목에 검색조건을 부여하면 프로그램은 검색조건이 모두 맞는 레코드의 데이터를 하나씩 화면에 표시한다.

또한 일반 데이터 검색에서는 같은 조건의 데이터를 한눈에 볼 수 없어 레코드를 비교할 수 없었는데 비해 한 화면에서 사용자가 검색하고자 하는 항목들을 직접 화면에 나열하여 레코드를 비교하며 검색할 수 있도록 되어 있다. 그리고 검색시에 데이터 수정을 할 수가 있으므로 사용자는 데이터 검색과 데이터 수정 기능을 동시에 이용할 수 있다.

• 데이터 출력

오아시스-V의 출력에서는 일반적인 출력과 함께 소계(SUB TOTAL)의 출력, 확대출력(Double Mode), 카드형태 출력(ll Mode) 등 다양한 형태를 제공할 뿐 아니라 일반 회사에서 필수적으로 사용하는 장부의 형태를 자유롭게 출력할 수 있도록 양식출력 기능을 제공하여 주므로 사용자는 다양한 형태(FORM)를 출력할 수 있다.

소계의 출력은 순차배열 항목순으로 사용자가 정의하여 사용하는 것으로 10개의 항목까지 지정할 수 있고, 지정시 중간에 공백이 있어서는 안된다는 점에 주의해야 한다.

(2) 일반 기능

• 레코드 정리

화일에서 불필요한 데이터를 삭제하고 레코드를 응축하는 기

능으로 다음과 같은 것을 행한다.

① 삭제표시 레코드 정리
② 검색으로 레코드 정리
③ 삭제표시 레코드 보기
④ 검색으로 레코드 복원

• 인덱스 정의

화일간의 연결 및 데이터 검색을 할 때 사용한다. 기능은 다음과 같다.

① 마스터화일과 일반화일의 인덱스를 구분하여 정의
② 마스터화일의 인덱스를 정의해 놓으면 데이터가 업데이트될 때마다 자동으로 인덱스 처리
③ 인덱스 취소기능
④ 일반 PC에서 사용해온 ISAM 방법이 아닌 유일한 방법으로 속도와 메모리 최대 활용

• 순차 배열

데이터 출력시 오름차순/내림차순으로 배열하는 기능이다.

① 10항목까지의 순위 지정
② 부분적인 데이터 소트기능
③ 지금까지 PC에서 사용되어온 소트유틸리티를 사용하지 않고 자체 제작한 특수 프로그램으로 기존 소트프로그램에 비해 처리속도 및 능력이 2배 정도 뛰어남

• 데이터 전송

요구되는 화일로 데이터를 전송하는 기능이다.

① 사용자가 전송받는 항목을 추가하여 계산식을 적용할 수 있음

• 자동 재계산

입력된 데이터를 요구대로 재계산하여 데이터의 이월작업이나 데이터의 이동, 데이터의 초기화 작업 등에 이용한다.

• 화일의 구조

화일에 대한 정보를 분류하여 출력하는 기능이다.

① 정보에는 항목의 편집내용/입력프로그램 및 화일정리 프로그램의 계산식 제공
② 데이터는 랜덤화일의 구조를 가지며 사용자는 데이터 화일을 오픈하여 일반 랭귀지로 활용할 수 있다.

• WP/LOTUS

WP/LOTUS내에 있는 데이터를 불러 오아시스-V 프로그램내에서 활용할 수 있는 기능이다.

• 화일의 병합

화일의 데이터를 조합(Merge)하여 새로운 화일을 재생할 때 사용하는 기능이다.

앞으로의 방향

그동안 데이터베이스분야에서 dBASE 시리즈나 PFS같은 프로그램이 국내 사용자들에게 많은 공헌을 해왔고 가장 문제시되던 한글도 dBASE에서 어느 정도는 지원해 주고 있는 상황이지만 이것은 단지 기존의 소프트웨어에 한글기능을 올려 놓았을 뿐 순수한 한글 패키지는 아니다. 그에 반해 오아시스-V는 순수 한글 패키지이기 때문에 그 의미는 깊다. 그러나 앞에서도 언급했듯이 삼보한글카드를 구비하여야만 이 프로그램을 사용할 수 있다는 점은 못내 아쉬움을 남긴다.

여하튼 오아시스-V는 누구나 쉽게 사용할 수 있도록 하는데 주안점을 둔 다기능 프로그램이고 사용자가 프로그램 자체를 이끌어 갈 수 있도록 한 업무용 데이터 베이스이다.

엘렉슨은 현재 삼보컴퓨터의 전문 대리점으로 아직 시스템이 구비되어 있지 않은 업체가 시스템을 구입할 경우에는 이 프로그램을 함께 보급하고 있으며 이미 구비되어 있는 경우에는 프로그램만 따로 판매하고 있다.

판매예정가격은 1백만원으로 책정되어 있다. 자세한 문의사항은 엘렉슨 교육부 이상백과장(전화 : (02)393-8030) 앞으로 하면 된다. PC

학원교육과정

글·이 향선기자

최근들어 정보화 사회의 분위기 조성과 함께 학부모와 학생들 사이에서 컴퓨터에 대한 관심이 고조되고 있다. 이에 따라서 80년 초에 일었던 컴퓨터 붐이 다시 일고 있는 실정이다.

이러한 제2의 컴퓨터 붐은 폭발적인 컴퓨터교육의 수요를 불러 일으키게 되었고 이 수요에 따라 컴퓨터학원이 급속도로 증가하게 되었다. 그리하여 현재 전국적으로 산재한 컴퓨터학원은 정확한 숫자를 셀 수 없을 정도로 난립되어 있으며 서울 시내만 해도 약 500여개의 학원이 난립되어 있다. 이러한 학원은 대체적으로 규모상 대형학원과 소형학원으로 나눌 수 있다.

대형학원은 범용 컴퓨터를 보유하고 정규과정을 운영하며, 주로 취업위주의 일반인들을 대상으로 교육하고 있다. 이에 비해 소형학원은 주로 8비트나 16비트의 컴퓨터를 보유하고 베이직이나 S/W 패키지를 교육하고 있다. 이 소형학원은 컴퓨터학원의 대부분을 차지하고 있으며 운영상태가 대체로 영세하다.

그리고 대부분의 컴퓨터학원이 교육목적을 사회교육기관으로서 국민자질 향상의 목적보다는 취업 등의 실용적인 목적에 더욱 치중하고 있다.

컴퓨터학원의 증가는 컴퓨터마인드 조성에 일익을 담당하여 컴퓨터를 접할 수 있는 환경을 확대하였지만 소형학원의 난립과 함께 교육할 강사의 부족현상, 시설의 미흡, 교육과정의 부실, 난립에 따른 수강료의 덤핑공세로 인한 상도덕의 문란과 같은 문제점이 대두되면서 컴퓨터 교육의 일면을 어둡게 하고 있다.

또한 학원이 지닌 독창성으로 인해 교육과정이라든지 변질된 형태의 운영에 대해 제재를 가할 수 있는 압력기관이 존재하지 않는다는 것이다. 이것은 관할 구청이 법적인 규제 능력이 있다 하더라도 수많은 학원을 감찰하지 못하고 있기 때문이다.

자질을 갖춘 강사의 확보 어려워

관계기관의 조사에 따르면 87년 6월 컴퓨터 교육기관당 평균 강사요원은 기관소속 강사가 11.1명, 외래강사가 5.5명으로 총 16.6명에 이르고 있으며 이 숫자는 85년 말의 14.3명보다 15.3%가 증가하였음을 보여주고 있으나 표 1에 나타난 바와 같이 우리나라의 정보산업을 주도할 전문인력의 수급 현황은 양적인 부족은 물론이고 시설과 교육기관, 또한 교수 인력의 질적인 부족현상에 부딪치고 있다. 이러한 문제의 파급은 관련기관이나 업체에서의 전문인력 부족은 물론 학원가의 전문강사 부족현상으로 나타나고 있는 것이다.

그리하여 대형학원의 경우 20명 이내의 강사가 1000~1400여명의 수강생들을 시간당 50~60명씩이나 지도하고 있는 실정이고 소형컴퓨터학원의 경우는 1개 학원당 수강생에 따라 2~7인 정도의 강사가 전체 인원을 지도하고 있다. 교육시간에 있어서는 소형학원의 경우 학교의 수업이 일정하게 끝나기 때문에 오후 4시에서 6시가 수강생이 가장 많은 시간대라고 한다. 이때에는 어쩔 수 없이 1인의 강사가 많은 수강생을 교육해야 하므로 교육효과가 낮아질 수 밖에 없다.

학원의 교과과정중 실습을 위주로 하고 있는 학원에서는 1인의 강사가 20~40명의 수강생을 1시간의 교육시간으로는 모두 교육하지 못하고 있어 초보의 수강생들은 단지 시스템의 조작법을 익히는 것으로 시간을 보내고 있는 실정이다.

강사의 부족현상에서 비롯되는 또 다른 문제는 자질있는 강사의 확보에 있다. 관계기관의 조사에 따르면 컴퓨터 교육기관의 강사중에서 대학원을 졸업한 사람은 40.2%, 대학교 졸업자가 27.3%이고 나머지는 전문대학 졸업자가 주류를 이루고 있다. 특히 외래강사에는 대학원졸업자가 많은 반면에 전임강사에는 대학졸업자와 전문대학 졸업자가 많은 특징을 갖는다.

전공별 구성을 보면 관련분야인 전자계산·정보처리학 전공자가 45.1%, 전기·전자공학이 7.1%, 통계·수학이 4.1%로 컴퓨터관련 전공강사는 전체의 56.3%를 차지하고 있다. 그리고 강사의 자격증 취득면을 볼 때 기술사, 기사 1·2등급 자격증을 취득한 강사가 66.3%로 나타나고 있고 정보처리

〔표 1〕 정보통신산업 종사인력 수급 예측

	87	92	97
수 요	16,553	26,952	38,367
공 급	11,739	19,222	26,719

문제점 많아

분야에서 기술사는 73.9%, 기사 1급은 79%, 기사 2급은 89.3%를 차지하고 있다(표 2, 3 참조).

그러나 이러한 통계는 전체 컴퓨터 교육기관에 관한 자료이므로 일반 소형 컴퓨터학원 강사의 자료와는 약간 차이가 있다.

대형학원의 경우 강사의 자격에 있어서 정보처리 기사 1급의 자격을 가진 전산관련학과 졸업생으로 1년이상의 실무 경력을 지닌 사람에 한해 강사의 자격이 주어지고 있는 반면 대부분의 소형학원 강사는 전산을 전공한 사람보다는 비전공자로 전문교육기관의 교육과정이나 대형학원의 전산 교육과정을 이수한 사람이 대부분으로 실무경험이 부족한 강사들이 교육을 담당하고 있는 실정이다.

소형학원의 경우 대부분 전산을 전공한 사람보다는 비전공자로 실무경험이 부족한 강사들이 교육을 담당하고 있는 실정이다.



물론 고등학교만 나오든지 대학을 나오든지의 문제는 접어 두고라도 가장 중요한 것은 교수법의 방법에 문제가 있는 것이다. 컴퓨터교육의 목적 자체가 컴퓨터 마인드 조성에 있다면 적어도 컴퓨터가 사회에 끼치는 영향이나 그 활용과 컴퓨터 문맹의 퇴치에 관한 내용을 인간관계에 적용시키는데 그 의미가 있다.

이러한 면에서 볼 때 고등학교를 졸업하고 컴퓨터의 원리적인 이해와 지식없이 몇개월과정의 전산교육기관에서만 받았던 컴퓨터지식으로는 컴퓨터 명령어를 가르칠 수는 있어도 교육과정과 관련된 컴퓨터의 올바른 개념 전달을 할 수 없기 때문이다.

특히 소형학원의 경우 수강생 중 국민학생이 많은 부분을 차지하고 있는데 이들에게 컴퓨터교육을 실시함에 있어 100% 컴퓨터지식만을 전달함에 그 목적을 두어서는 안될 것이며 컴퓨터에 관한 흥미유발 및 컴퓨터를 통해 창의력을 개발하도록 하여 다른 교육에도 미칠 영향을 고려한 교수법이 필요하기 때문이다

이러한 면에서 대부분의 컴퓨터학원은 아직까지 컴퓨터에 관한 이론에만 치우친 주먹구구의 교육방식과 주입식 교육에 치우치고 있어 오히려 마인드 자체를 파괴하고 있는 실정이다.

체계적인 교수법이 시급

이와 같이 자질을 갖춘 강사의 부족 원인을 보면 대부분의 소형학원이 영세한 운영을 하고 있기 때문에 비롯되

〔표 2〕 교육기관 강사요원의 전공 및 학력(1기관 평균인원) (단위 : 명, %)

전공분야 \ 학력별	대학원졸	대학졸	전문대졸	고교졸	기 타	합 계
전자계산·정보처리	3.1 (69.2) 〈61.7〉	1.1 (41.3) 〈21.7〉	0.6 (71.4) 〈12.5〉	–	0.2 (9.1) 〈4.2〉	5.0 (45.1)
전 기·전자공학	0.3 (6.5) 〈36.8〉	0.4 (14.3) 〈47.4〉	0.1 (14.3) 〈15.8〉	–	–	0.8 (7.1) 〈100.0〉
통 계·수 학	0.2 (4.7) 〈45.5〉	0.3 (9.5) 〈54.5〉	–	–	–	0.5 (4.1) 〈100.0〉
경 영·경 제 학	04. (9.3) 〈58.8〉	0.2 (9.5) 〈35.3〉	0.1 (4.8) 〈5.9〉	–	–	0.7 (6.4) 〈100.0〉
기 타	0.5 (10.3) 〈11.1〉	0.7 (25.4) 〈16.2〉	0.1 (9.5) 〈2.0〉	0.8 (100.0) 〈20.2〉	2.0 (90.9) 〈50.5〉	4.1 (37.2) 〈100.0〉
합 계	4.5 (100.0) 〈40.2〉	2.6 (100.0) 〈23.7〉	0.9 (100.0) 〈7.9〉	0.8 (100.0) 〈7.5〉	2.3 (100.0) 〈20.7〉	11.1 (100.0) 〈100.0〉

주 : ( )는 전공별 구성비, 〈 〉는 학력별 구성비임.
자료 : 한국정보산업연합회

〔표 3〕 강사의 자격증 취득현황 (1기관 평균인원) (단위 : 명, %)

기술분야 \ 등급		기 술 사	기사 1 급	기사 2 급	자 격 증	합 계
정보처리	정 보 처 리	1.6 (73.9) 〈32.7〉	1.5 (60.7) 〈32.7〉	1.7 (64.3) 〈34.6〉	—	4.8 (43.7) 〈100.0〉
	수 학 응 용	— —	—	—	—	—
	전자계산기	0.5 (26.1) 〈10.9〉	0.3 (10.7) 〈5.5〉	0.6 (21.4) 〈10.9〉	3.7 (100.0) 〈72.7〉	5.1 (46.2) 〈100.0〉
	조 직 응 용	—	0.1 (3.6) 〈50.0〉	0.1 (3.6) 〈50.0〉	—	0.2 (1.7) 〈100.0〉
전 자		—	0.3 (10.7) 〈60.0〉	0.2 (7.1) 〈40.0〉	—	0.5 (4.2) 〈100.0〉
전 기		—	0.1 (3.6) 〈50.0〉	0.1 (3.6) 〈50.0〉	— —	0.2 (1.7) 〈100.0〉
기 타		—	0.3 (10.7) 〈100.0〉	—	—	0.3 (2.5) 〈100.0〉
합 계		2.1 (100.0) 〈19.3〉	2.6 (100.0) 〈23.5〉	2.6 (100.0) 〈23.5〉	3.7 (100.0) 〈33.6〉	11.1 (100.0) 〈100.0〉

주 : ( )는 분야별 구성비, 〈 〉는 등급별 구성비
자료 : 한국정보산업연합회

며 그로 인해 파급되는 강사의 대우 문제로 지적된다고 볼 수 있다.

대부분의 컴퓨터 학원이 전산관련의 타직종과 같이 고정적인 수입원을 이루지 못하고 있어 일부의 학원에서는 강사에게 분할제를 적용하여 수강생의 다소에 따라 임금을 책정하는 등 강사의 임금이 고정적이지 않거나 고정급으로 책정이 되었어도 일반 직종의 임금에 비해 낮기 때문에 강사의 유동이 잦다. 이러한 강사의 유동은 교과과정에 변화를 일으키는 등 학원운영상의 문제점을 야기시키고 있다. 그러므로 강사의 유동에 대한 대책으로 학원경영주는 고임금의 자격을 갖춘 강사보다는 저임금 인력을 채용하고 있다.

그 결과 전문적인 자질과 실무를 접하지 않은 강사가 교육함으로써 실질적인 학원교육의 질적인 저하를 초래하고 있고 수강생들의 대부분도 처음에는 흥미를 가지고 수강하게 되지만 2내지 3개월 정도만 되면 곧 흥미를 잃고 무조건 어렵다고만 느껴 도중에 포기하는 경우가 절반이 넘는다. 실제 학원의 6개월과정에서 끝까지 남아있는 수강생은 전체의 40%정도에 지나지 않는다고 한다.

그러면 강사의 자질을 향상시키려면 어떻게 해야하는가 그에 대하여는 관련단체의 자체적인 정비와 자질향상을 위한 교육의 증가 및 시설 확충 그리고 컴퓨터교육에 대한 체계적인 교수법의 개발이 촉구된다.

특히 전산요원의 교육에 있어서 이론과 연계를 가진 실무교육의 강화 및 교육내용과 교수법에 관한 교육체제를 확립하는 등의 대폭적인 개선이 요구된다.

학원관계자의 말에 따르면 학원 강사에게 필요한 것은 컴퓨터에 관한 해박한 지식 및 경험이겠지만 가장 필요한 것은 어떻게 컴퓨터 지식을 수강생에게 전달하느냐에 있으므로 그러한

방법에 관한 체계적인 강사 교육기관의 제도 마련이 시급하다고 한다. 그래야만 정보화시대가 요구하는 올바른 교수법이 정립되어 강사의 자질을 향상시킬 수 있을 것이다.

대부분이 학원시설 기준에 미달돼

대부분의 교육기관들이 시설부족으로 인해 충분한 교육이 이루어지지 않고 있다. 특히 컴퓨터 학원에서는 타분야보다는 실습이 차지하는 비중이 높아 시설설비에 많은 투자가 필요하다.

문교부지정 컴퓨터학원의 시설기준을 보면 실습실 66㎡ 이상의 면적에 퍼스널컴퓨터 과정은 주변기기를 갖춘 퍼스널컴퓨터를 일시 수용인원 1인당 1대의 시설로 설치해야 하고 키펀치의 과정은 입력작성을 위한 기기가 1인당 1대를 사용할 수 있도록 갖추어야 하며 한반의 정원은 40명으로 되어 있다.

조사에 의하면 시스템의 설치현황은 평균적으로 1개 기관당 27.7대로 되어 있으며, 이중에는 PC가 24.9대로 전체의 89.9%, 마이크로급이 2.0대로 7.6%, 소형과 중형이 0.3대, 대형과 초대형이 0.1대로 소형이상의 기종은 거의 드물게 보유하고 있는 실정이다(표 4 참조).

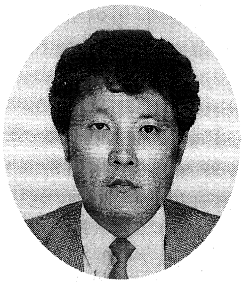
이러한 컴퓨터시설의 보유는 85년말

이론과 연계를 가진 실무교육의 강화 및 교수법에 관한 교육체제 확립이 전산요원 교육에 있어 가장 시급한 문제이다.

●인터뷰●

통일된 교재 있어야 체계적인 교육이루어져

박병기/인천 중앙컴퓨터 학원강사



"요즘 부족한 시설과 자질을 갖춘 강사의 부족으로 인해 부실한 교과과정을 실시하고 있는 영세학원이 난립함에 따라 컴퓨터 교육자체가 위협받고 있는 것은 사실입니다. 그러나 이러한 어려움을 해결하기 위해서는 학원측의 투자기반 확립도 필요하겠지만 그에 따른 관계기관의 재정지원 및 학원 강사를 위한 연수, 교육기관제도 등이 마련되어야 합니다"라고 말하는 인천 중앙컴퓨터학원 강사 박병기씨.

강사로서의 애로점에 대하여는 "체계적이고 통일된 교재가 마련되어 있지 않아 컴퓨터교육의 연계성을 이루기 어렵습니다. 그러므로 관계기관에서 통일된 교재를 선정하여 일관성있는 교육이 이루어지도록 하는 방안을 마련해야 할 것입니다"고 하며 정부의 컴퓨터교육 육성책인 교과과정의 컴퓨터교육 도입은 컴퓨터교육을 담당하는 사람으로 환영할 일이며 이러한 정책이 어느 정도 컴퓨터교육의 방향정립에 도움이 되었으면 한다고 한다. 그리하여 학원과 학교가 서로 연계된 교육을 실시한다면 컴퓨터교육은 성공할 것이라고 내다보고 있다.

그리고 학원 난립에 따라 문제시 되었던 것들에 대해서는 학원의 자체정비와 아울러 학원경영의 내실화를 기할 수 있는 관계기관의 재정 및 정책적인 뒷받침이 필요하다고 보고 있다.

또한 학원의 자체정비면에서는 "지금까지 쌓아왔던 경험과 기술을 바탕으로 새로운 교수법을 개발함과 동시에 전문적이고 실질적인 교육을 강화하여 그동안 이론위주로 실시되어 실무 적용이 어려웠던 점을 극복하여야 합니다. 그러므로 실습교육을 강화하면서 컴퓨터의 전반적인 흐름을 이해할 수 있는 교과과정을 위주로한 교육이 실시되어야 합니다"라고 지적하고 있다.

"컴퓨터를 배우는 사람들중 일부에서는 남들이 배우니까 나도 배워야 한다는 식의 마구잡이 교육태도는 시정되어야 할 것입니다. 특히 학생들을 지도하다 보면 중간에 포기하는 중도탈락율이 매우 높습니다. 그러므로 컴퓨터를 좋은 친구로 만드는 것은 포기하지 않는 인내력이라고 생각합니다"라고 수강생의 의욕상실도 교육상 문제점 중의 하나로 지적하는 박병기씨의 말에서 문제는 어느 한편에 만 있는 것이 아니며, 참다운 정보화 시대로 가기위한 거보는 모두의 노력에 있음을 알 수 있다.

〔표 4〕 컴퓨터교육기관의 컴퓨터 설치현황(1기관 평균대수) (단위 : 대, %)

시점 / 형별 \ 도입형태	1985년말 구입	1985년말 임대리스	1986년말 구입	1986년말 임대리스	1987년 6월말 구입	1987년 6월말 임대리스
초대형	—	—	0.1 (0.4)	—	0.1 (0.4)	—
대형	0.1 (0.5)	—	0.1 (0.4)	—	0.1 (0.4)	—
중형	—	—	0.2 (0.9)	—	0.3 (1.2)	—
소형	0.1 (0.5)	0.2 (0.9)	0.1 (0.4)	—	0.3 (1.2)	—
초소형	0.1 (0.5)	0.1 (33.0)	0.2 (0.9)	0.3 (30.0)	0.2 (0.8)	1.8 (100.0)
P C	20.8 (98.6)	0.1 (33.0)	21.9 (97.7)	0.7 (70.0)	24.9 (96.1)	—
계	21.1 (100.0)	0.3 (100.0)	22.6 (100.0)	1.0 (100.0)	25.9 (100.0)	1.8 (100.0)

주 : ( )내는 구성비임.
자료 : 한국정보산업연합회

21.4%에서 86년말 23.6%, 87년말에는 27.7%가 증가하였다. 따라서 85년에 비해 17.4%가 증가된 점을 고려해 볼 때 보유대수도 늘어날 것으로 예측되지만 일반 컴퓨터학원에서는 아직도 시스템이 모자라 실습실에서의 수강생들의 자리다툼이 끊이지 않고 있다. 비근한 예로 어느 학원에서는 시스템의 부족으로 인해 시스템을 한 사람이 독점해서 사용하다 보면 여러 사람이 사용할 수 없기 때문에 궁여지책으로 1시간의 사용료를 지불하게 하고 있다고 하는데 전문 인력을 양성하는 학원에서의 아이러니가 아닐 수 없다.

이것은 소형학원에서도 마찬가지로 1인이 1대의 시스템으로 실습해야 하는데 1대에 2인 이상이 사용하므로 개인의 사용시간이 제약받고 있다. 그리하여 학원 수강생의 대부분이 실습시간의 부족을 학원측에 호소하고 있는데 학원 측에서는 운영상의 이유와 교육방침이 키펀치 과정이 아님을 밝히면서 시스템 접촉시간과 프로그램 작성관계는 별도의 문제로 취급하고 있는 곳도 있다.

이러한 문제점은 학원측이 고가의 시스템을 구입함에 있어서 임대, 리스보다는 구입에 의존하기 때문에 더 큰 문제로 작용되고 있는 실정이다.

이러한 면에서 볼 때 학원 경영자는 컴퓨터교육의 내실화를 기하고 교육과정을 다양화하기 위하여 주교육교재가 되고 있는 컴퓨터시스템과 주변기기의 충분한 설치, 컴퓨터시스템의 빈번한 신기종 출하에 대응한 현실적인 투자를 하여야 할 것이다.

그러나 컴퓨터 교육기관 중에서 흑자를 내고 있는 기관이 불과 26.6%에 지나지 않은 점을 감안할 때 특히 미약한 자본으로 경영되고 있는 소형 컴퓨터학원의 자력투자의 여건은 극히 취약한 상태이다.

그러므로 미래의 전산전문인력 증대를 목적으로한 교육기관에서는 이를 대비한 높은 수준의 전문 교육과정을 신설하여 효율적인 운영을 꾀하여 중형급의 시스템을 운용하는 것이 더 바람직한 현상이며 이러한 대형·소형학원의 시설 설치에 대한 재정지원에 관하여 관계기관의 배려가 요구된다.

체계적인 교육이 이루어져야

그동안 대부분의 컴퓨터 교육은 사설강습소가 담당하여 정보산업사회를 이끌어 갈 인력들을 육성해왔다.

그러나 컴퓨터마인드의 정착을 위한 정부의 발표와 함께 사설학원들이 갑자기 많이 늘어나 영세학원이 난립하기 시작하였으며 이러한 난립은 기존 컴퓨터 학원이 가졌던 컴퓨터 교육담당기관으로서의 위치를 위협하면서 그간 다져온 학원으로서의 고유영역과 그 가치성을 서서히 흐리고 있다.

이러한 원인은 학원이 양적인 팽창에만 주력하였을 뿐 교육과정에 있어서 체계적이고 합리적인 방법의 채택을 등한시하였기 때문이다. 그리하여 교육의 다양화와 수준높은 교육내용의 부재현상이 나타나고 있다.

관계기관의 조사에 따르면 대부분의 컴퓨터교육기관에서 프로그래밍과정을 실시하고 있으며, 키펀치과정과 시스템 분석과정을 실시하고 있는 곳은 각각 34.8%와 30.4%에 불과하다.

그리고 강의시간은 시스템분석과정이 평균 364.3시간, 실무과정 298.8시간, 프로그래밍과정 239.3시간 등이며, 주제별 세미나가 12.3시간, 관리자과정이 20.0시간으로 되어 있다.

교육과정의 수를 보면 1개 기관당 평균 2.8개로 85년 이후 거의 변동사항이 없는 반면 수강인원은 86년이 전년대비 63.3%, 87년 상반기에는 연율로 52.0%가 증가함으로써 일반인의 컴퓨터에 관한 관심도도 증가하고 있음을 알 수 있다. 그러나 교육기관에서의 연간, 주당 개강일수가 변동이 없는 상태임을 감안할 때 수강생이 양적으로 늘어나는데 비해 교육과정의 신설 등은 이루어지지 않고 있는 상태이며 현시설로 늘어나는 수강생을 수용하고 있음을 시사하고 있다(표 5 참조).

그리고 기관당 평균 2.8개의 교과과정이 운영되고 있는데 61.9%를 차지

〔표 5〕 컴퓨터교육과정 운영실적(1기관 평균)

항목 \ 년도별	단위	1985년	1986년	전년대비 증가율(%)	1987년 1월~6월
운 영 과 정 수	개	2.8	2.8	0.0	2.8
총(연)수강신청인수	명	533.3	870.8	63.3	662.0
총(연)수료인원수	명	341.3	404.4	18.5	274.0
년 간 개 강 일 수	일	234.0	233.4	△0.3	123.3
주 당 개 강 일 수	일	5.5	5.6	1.8	5.2
1 일 개 강 일 수	시간	5.8	6.3	8.6	6.6

자료 : 한국정보산업연합회

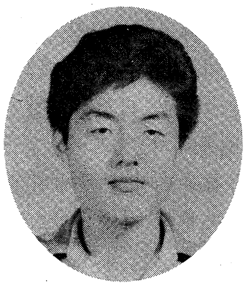
하고 있는 2개 이하의 교육과정을 개설하고 있는 기관들은 기초과정을 주된 교육과정으로 채택하고 있으며 나머지 38.1%를 차지하고 있는 기관에서는 중급과 고급과정을 포함하고 있다(표 6 참조).

컴퓨터교육의 내실화를 위해서는 교육과정의 다양화와 컴퓨터 시스템과 주변기기의 충분한 설치 등 현실적인 투자가 이루어져야 할 것이다.

● 인터뷰 ●

충분한 실습환경이 마련되어야 올바른 교육이루어져

김용준/인천대학교 산업공학과



"사실 수강생들은 이론보다 실제적인 시스템을 다루면서 컴퓨터에 익숙해 지길 원합니다. 그러나 시설이 부족하여 실습실이 항상 만원입니다. 그리고 실습시간에도 조를 짜서 진행하고, 시스템을 풀가동하다 보면 시스템의 잦은 고장으로 실습시간도 제대로 실습할 수가 없습니다. 거의 완전히 실습을 위주로 하는 COBOL시간의 고생은 더욱 많았습니다"라고 하는 인천대학교 재학생 김용준군은 정보화시대에 뒤떨어지지 않기 위하여 남보다 먼저 컴퓨터를 이해하고자 컴퓨터학원에 등록하였다고 한다.

학원에 수강하고 나서 수업이 진행됨에 따라 컴퓨터에 흥미를 느끼기 시작했다는 그는 먼저 컴퓨터에 대한 전반적인 이론을 배우고 난뒤 언어교육과정에 들어와서 실습시간을 가지게 되었다고.

"대부분의 수강생이 실습시간이 부족하여 컴퓨터학습에 어려움을 겪고 있으므로 학원이 시설확보에 더 많은 투자를 함으로써 충분한 시스템 환경을 구비하여 수강생들이 시스템 앞에서 겁먹지 않고 당당하게 시스템을 다룰 수 있었으면 합니다. 그리고 초보자에게 좀더 쉬운 교재가 준비되어 있어 컴퓨터를 이해하는데 거부반응을 일으키지 않고 시작할 수 있도록 하였으면 합니다. 때로는 선생님들의 강의가 잘 이해되지 않을 때도 있습니다."라고 하는 김용준군의 바램이 비단 그의 개인적인 의견이 아님을 볼 때 시설확충을 위한 학원 경영주의 자성과 함께 정부의 투자기반 마련책, 그리고 체계적이고 올바른 교수법의 확립이 가장 시급한 문제가 아닐까 생각해 본다.

〔표 6〕 교육과정수별 개설기관수

구분 \ 교육과정	1개과정	2개과정	3개과정	4개과정 이상	합 계
개설기관수(개)	5	8	6	2	21
비 중(%)	23.8	38.1	28.6	9.5	100

자료 : 한국정보산업연합회

그러나 학원의 운영상태는 그 규모에 따라 교과과정에서 현격한 차이를 빚어내고 있는데 우선 범용의 컴퓨터를 보유하고 있는 학원의 경우 표 7에 나타난 바와 같이 정규과정과 특별과정으로 나뉘어 실시하고 있고 특별히 정규과정을 모두 수료한 이후에는 수료증이 발급되기도 한다.

대형학원의 경우 많은 수강생을 바탕으로 어느정도 안정된 교육과정을 실시하고 있으나 문제는 난립하고 있는 소형학원의 교육과정이라 할 수 있는데 소형학원의 경우는 교과과정 자체가 전문성을 띄고 있지 않아 누구를 주된 대상으로 체계적인 교육을 실시해야 할지 그 방향설정이 모호하다는 것이다.

소형 컴퓨터 학원의 경우 대부분 국민학교 3학년부터 수강신청을 받고 있는데 5, 6학년이 대부분인 이들을 대상으로 학원에서는 주로 베이직을 가르치고 있다. 그러나 베이직도 조금 더 깊이 들어가게 되면 수학적인 함수들이 등장하게 되어 수강생이 이해하는데 힘들어 한다고 한다. 그래서 맨처음 흥미와 열의를 가졌던 수강생들은 점차 흥미를 잃고 포기하게 된다. 물론 수강생의 노력부족이라는 이유도 있겠지만 가장 크게 좌우하는 것은 교수법의 차이에서 비롯된다고 보고 있다.

그리고 단계적이고 통일적인 교재가 마련되지 않아 학원에서 교재를 선택할 때도 어려움을 겪고 있다고 한다. 이것은 컴퓨터교육이 어느 정도 통일

〔표 7〕 대형 컴퓨터학원의 교육과정

정규과정	전산학 개론	4주
	포트란	4주
	코볼 I	4주
	코볼 II	4주
	어셈블리	4주~8주
	OJT	4주
특별과정	키펀치과정	16주
	기사특강과정	4주
	위탁교육과정	
	OA과정	
	패키지사용과정	

된 교과과정이 없이 학원 임의로 정하여 이루어지고 있으며 교육의 방향자체가 프로그래밍 작성법에만 국한하고 있기 때문이다.

이러한 교육과정의 부재현상이 컴퓨터 마인드조성의 저해요인으로 등장하자 정부에서는 제5차 교육과정개정을 통하여 국민학교, 중학교, 고등학교 교과과목에 컴퓨터에 관한 부분을 첨가 혹은 신설하도록 지시한 바 있다. 컴퓨터교육방향이 제시되긴 하였지만 이것은 학교의 컴퓨터교육에 관한 것으로 물론 학원의 교육 내용에는 영향을 크게 미치지는 않을 것이다. 그러나 컴퓨터 교육을 담당하는 기관으로서 학교와 연계성을 지닌 교육을 실시한다면 교육효과를 높일 수 있을 것으로 교육관계자는 내다보고 있다.

학부모들도 학교의 교과과정과 유리된 컴퓨터교육은 오히려 학습의욕을 저하하지 않을까 하는 우려를 하고 그

소형학원의 경우 교과과정 자체가 전문성을 띄고 있지 않아 누구를 주된 대상으로 하는지 방향설정이 모호하다.

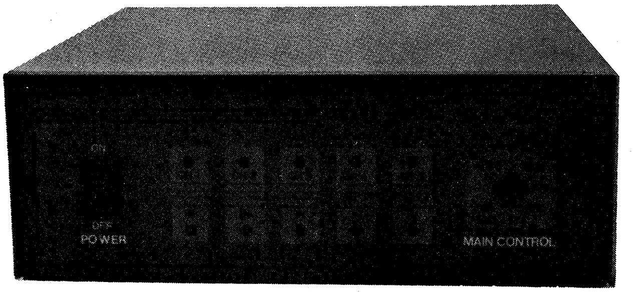
에 대한 개선책을 요구하고 있다. 이러한 점에서 볼 때 학원관계자 및 학부모, 교육관계자들은 교과지도용 컴퓨터 프로그램인 코스웨어(Course Ware)가 다양한 부분에서 개발되어 교육의 능률 및 컴퓨터 인식을 높여야 할 것이며 적절한 교재가 지정되어 연계성을 띠는 교육이 이루어져야 한다고 보고 있다. 또한 CAI, CMI, 비디오, 슬라이드 등을 교육의 보조도구로 활용하여 특성있는 교육의 실시 방안이 시급하다고 보고 있다.

미래는 정보화 시대임을 누구나 직감하고 있는 현실에서 컴퓨터의 올바른 인식을 위한 교육은 우리의 미래에 대한 계획임을 직시하고 컴퓨터 교육의 통일된 교육관 정립과 그에 따라 국가적인 차원의 지원이 함께 하여 누구나 만족한 정보사회가 이룩되어야 할 것이다. PC

PRINTER 주변기기 전문 메이커

- 컴퓨터는 2대 혹은 3대이고, 프린터는 1대뿐이라 불편하게 사용하고 계십니까?
- 컴퓨터 1대에 프린터를 1대씩 사용하거나 추가 구입시 프린터를 구매하려고 생각하십니까?
- 프린터를 하고 있는 동안 컴퓨터를 장시간 사용하지 못하고 계십니까?
- CAD 사용시 Plotter가 그리고 있는 동안 컴퓨터를 장시간 사용하지 못하고 계십니까?
- 컴퓨터의 수량과 프린터의 수량이 안맞아 적절한 사용을 못하고 계십니까?
- 학교 및 학원의 실습실에 프린터대수가 부족해 적절한 교육을 못하고 계십니까?
- F.A 혹은 계측기쪽의 프린터 인터페이스에 문제가 있읍니까?

컴퓨터 사용중에 불편한 위의 모든 문제를 자신있게 처리해 드리므로, 귀사의 업무에 도움이 되어 드리고 싶읍니다!



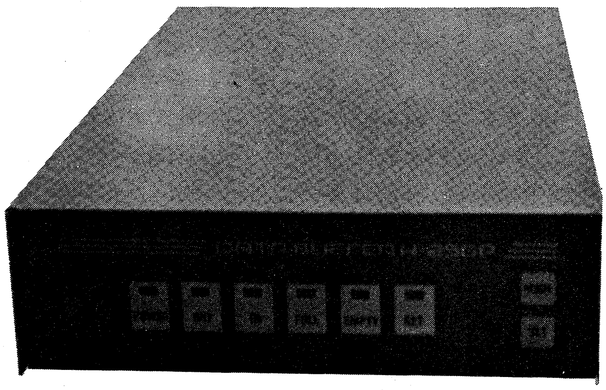
※ K-P-345A는 컴퓨터 프로그램 보호법에 의해 등록된 제품입니다.

K-P-345A의 사양

입·출력 방식	Centronics Type
입력 COMPUTER	3:1~5:1
출력 PRINTER	1대
대기 시간	5초, 10초, 20초, 40초 중 선택
채널 변환방식	전자식
채널 선택방식	자동 or 반자동
용지 F.F 기능	자동 F.F or None F.F
F.F 삭제 기능	자동 삭제 or None

K-256P의 사양

버퍼 용량	256kB(80col 약 200장)
입·출력 방식	Centronics Type
입·출력 수	1:1 or 2:1
사용자 MODE	10EA
고유 기능	Data 중 Space 입력 기능, 재 Copy 기능, 메모리 기능, Self Test 기능 등 총 10가지



DATA SWITCH
- 전자식 완전자동 (2:1~8:1 or 64:1)
- 전자식 반자동 (2:1~8:1, 4:2)
- Serial 완전자동(4:1, 8:1)

DATA BUFFER
- 소용량 64kB
- 고성능 다기능 256kB
- 대용량 다기능 1MB
- Serial용 1MB

DATA CONVERTOR
- Serial→PARALLEL
- PARALLEL→Serial

각급 학교 및 학원의 전산 실습실 프린터 공유장치 공사

ATTA SYSTEM
PRINTER 주변기기 전문메이커
서울·종로구 종로 3가 175-4 (현대APT1309호)
전 화: 2 7 7 - 1 7 3 9
F A X: 2 7 7 - 8 2 5 1

● 상기 제품은 아래의 전문 취급점에서 전시 판매하고 있으며 직접 사용해 보실 수 있읍니다.●

서울 : 세운상가 협진 274—5419(가동 라열 426호)
용산 나진상가 동해전자 719—8010(17동 가열 304호)
용산 전자랜드 협진 712—2764(3층 C-336호)

인천:바이너리 컴퓨터 (032)762—5544
대구:범진 컴퓨터 (053)252—2211
마산:한길 컴퓨터 (0551)94—5555
원주:중부 씨스템 (0371)43—7979
부산:한국컴퓨터씨스템 (051) 42—2048

(기타지역 전문 취급점 모집중)

신간안내

TURBO C의 모든 것 (확장편)

최인현 저/대림출판사 발행/
정가 7,000원/387면/4·6배판



C언어가 발표된 이후 C언어에 관한 많은 서적이 발간되었으나 C 언어의 본질적 속성만 소개되고 구체적인 특성에 관하여는 언급이 부족한 상황이었다.

본서는 터보 C의 기초부분을 이해하고 강도있게 학습하려는 중급 사용자를 대상으로 한 고도화된 사용방식에 비중을 둔 터보 C의 활용서이다.

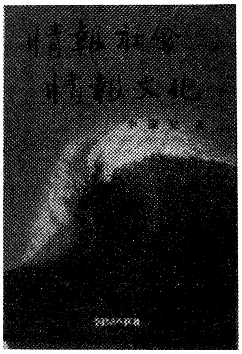
특히 C의 라이브러리 함수 사용에 관하여 라이브러리의 개요부터 화일의 입, 출력을 체계적으로 설명하여 라이브러리 함수 이용능력을 배가하는데 중점을 두고 있다.

또한 C언어로의 프로세스제어, 도스 인터페이스, 메모리할당, 메모리운영 등의 개념의 이해도 함께 설명하고 있다.

특히 C언어의 초심자는 이전에 발행된 〈기초편〉과 같이 학습하면 더욱 체계적으로 C를 이해할 수 있을 것이다.

정보사회 정보문화

이용태 저/정보시대 발행/
정가 4,500원/320면/국판



이 책은 정보화사회의 변화를 제시하고 이러한 변화의 물결을 어떻게 하면 순조롭게, 그리고 어떻게 하면 남들 보다 빨리 받아들일 수 있는가에 대한 방법을 제시하고 있다.

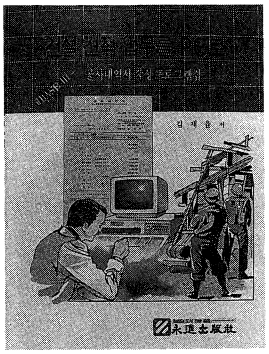
기업에서의 변화, 가정에서의 변화, 학교에서의 변화, 농·어촌에서의 변화등 우리의 모든 주변에 대해 다가오는 정보사회와 이 안에서 발생될 정보문화에 대해 확실한 답변과 함께 선진국으로 가는 방향도 제시하여 주고 있다.

이 책는 그동안 전문가 중심의 정보사회 관련책자와는 완전히 성격을 달리하여 남녀노소 누구라도 쉽게 느낄 수 있도록 120여개의 삽화를 곁들여 꾸민 에세이집 이다.

전문성을 떠나 가볍게 정보화사회의 이모저모를 훑어 볼 수 있는 책으로 권장할 만한 책이다.

건설견적업무를 PC로

김재율 저/영진출판사 발행/
정가 8,000원/263면/4·6배판



건설 기술에 있어 PC를 사용하는 기술자가 늘어가는 추세에 비해, 이에 부응하는 프로그램의 부족으로 PC를 보유하고 있는 회사들마저도 고가의 프로그램을 구비하지 못하여 몇몇 업무외에는 거의 컴퓨터가 사장되는 곳이 많다.

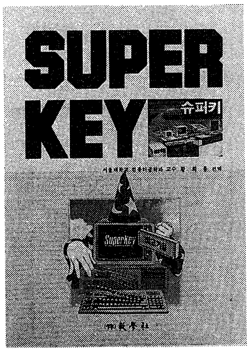
간단한 프로그램이라면 직접 PC 앞에 앉아 작성할 수 있을 것이다. 그러나 반복되는 업무를 볼 때마다 프로그램을 만들어 사용할 수도 없는 일이다.

서점가에서 계속 쏟아져(?) 나오는 컴퓨터 관련 서적을 아무리 펼쳐보아도 도저히 프로그램을 만들어보고자 하는 엄두가 나지도 않고, 그렇다고 컴퓨터 내부에 들어가 속속들이 파헤칠 수도 없고해서 시작도 하지 않고 포기해야 하는 기술인들이 많아지고 있는 실정은 감출 수 없는 현실이다. 그것은 컴퓨터가 건물을, 도로를 직접 건설하는 것이 아니기 때문이다.

이 책은 처음부터 끝까지 디베이스로 작성된 거대한 프로그램을 설명하고 있는데, 여기에 소개된 프로그램은 컴퓨터프로그램 보호법에 의해 과학기술처에 등록(등록번호 88-01-12-0417, 등록명칭 ESTIM-I, 등록일 1988년 9월 20일)된 건설분야 내역서 작성용 프로그램으로, 앞으로 건설분야의 기술축적을 위해 좋은 귀감이 되리라 본다.

SUPER KEY

황희융 편역/교학사 발행/
정가 4,500원/244면/4·6배판



수퍼키(Superkey)는 미국Borland사의 제품으로 키보드강화 유틸리티이다.

쉬운 예로 DOS프롬프트 상에서 주로 쓰이는 각종 명령어 라인을 하나의 키로 대치시킬 수 있으며 사용자 전용의 도움말 메시지화면 설계, 또는 과거에 사용했던 코맨드(Command)를 선택해서 다시 실행시키는 기능, 화일을 암호화시킬 수 있는 기능, 키보드 배열 재정의 등 키보드에 관한 온갖 고급 기능들을 고루 갖추고 있다.

이 책은 수퍼키의 사용법을 초보자일지라도 단계적으로 익혀나갈 수 있도록 단계별로 예제를 들어가면서 자세하게 설명되어 있다. 이의 구성을 보면 제 0 장에는 초보자를 위해 예제를 통한 수퍼키의 개략적인 사용법이 나와 있고, 제 1장과 2장에는 수퍼키를 설치하는 방법과 마무리 작업 및 간단한 사용법이 나와 있으며, 제 5 장과 6 장에는 에디터를 통한 수퍼키의 사용법과 스프레드 시트와 함께 수퍼키를 쓰는 방법이 나와 있다. 또한 6장과 7장에는 매크로(Macro)를 편집하는 방법과 키보드 배열에 관한 편집정보가 소개되고 있으며, 8장에는 수퍼키의 기능들이 알파벳 순서대로 정리되어 있다.

한편 부록편에서는 사이드킥과 함께 수퍼키를 이용하는 방법과 배치화일의 이용법들이 나와 있다.

한글 영문 한자 워드프로세서
보석글 Ⅱ 활용

편집부 편저/영진출판사 발행/정가 4,000원/146면/4·6배판



이 책은 워드프로세서 소프트웨어중 국내에 가장 많이 보급되어 있는 보석글 Ⅱ Ver2.0에 대한 활용 입문서이다.

흔하들 보석글은 기능이 너무 많아 도리어 이용하기가 불편하다고 한다. 한 예로 한가지 기능을 얻기 위해서 메뉴선택 방법과 명령어 입력 방법, 그리고 기능키 이용법등 3가지나 존재할 정도로 그 이용법이 다채롭다. 이 책에서는 이런 풍부한 기능들을 삽화와 각 명령별로 관련된 화면을 그대로 옮겨 이해가 쉽도록 구성되어 있다.

책 구성은 전체 11장으로 되어 있는데 1장과 2장에서는 보석글Ⅱ의 전반적인 개요와 실행하기 전에 알고 있어야 할 사항들에 대해 기술되어 있다. 3장과 4 장에서는 문서의 작성과 편집방법에 대해서 설명되어 있고, 5장에서는 편집된 문서내에서 자료를 옮기는 기법에 대해서 설명하고 있다.

그리고 6장에서는 검색과 대치에 관하여, 7장은 작성된 화일의 효율적인 관리방법에 대해, 8장은 여러가지 문자체를 선택하여 인쇄하는 방법이 실려 있다.

9장과 10장에는 여러가지 문서 작성의 예가 나와 있고, 끝으로 11장은 부록편으로서 워드프로세서 사용자를 위한 조언과 보석글Ⅱ의 종류와 명령어, 한자처리 방법, 워드프로세서에 관한 용어해설 등이 수록되어 있다.

한편 보석글Ⅱ의 스프레드 시트, 데이터 베이스, 그래픽 기능에 대해서는 이 책에서 다루고 있지 않다.

알기쉽게 풀이한
BASIC 프로그래밍정석

이종락외 1인 공저/교학사 발행
정가 5,000원/241면/4·6배판



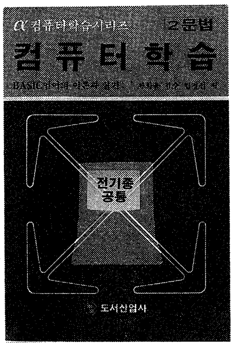
이 책은 저자 나름대로 그동안 전산교육을 담당하면서 얻은 경험을 토대로 베이직 프로그래밍 학습에 도움을 주고자 만든 책이다.

이 책의 특징을 몇가지로 나눠보면 첫째로, 학생 스스로 학습하는데 이해하기 쉽도록 예제와 해설이 많이 나와 있으며 둘째로, 예제를 쉬운 것부터 어려운 것의 순서로 배열하여 단계적인 학습과정을 거쳐 응용력을 기르도록 했으며 세째로는 호환성을 고려하여 가능한 한 국내에서 사용되는 모든 기종에서 사용할 수 있는 명령어로 프로그램을 작성하였다는 점이다.

전반적인 책 구성을 보면 제1장과 2장에는 PC의 사용법과 BASIC의 기초가 나와 있고, 3장에서부터 7장까지는 BASIC의 기본 명령어와 입출력문, 제어문, 함수등이, 8장에서부터 11장까지는 반복 처리문, 배열처리, 부프로그램 작성요령, 탐색과 분류에 대해서 나와 있다.

α컴퓨터학습 시리즈 2
컴퓨터 학습

임경선 저/도서산업사 발행/정가 4,000원/298면/국판



이 책은 저자인 임경선씨의 10년 이상되는 실지 산교육 (어린이 및 성인 컴퓨터 교육) 경험을 바탕으로 이루어진 책이며, 반복된 교육과정에서 배우는 사람들의 심리를 파악하여 싫증을 느끼지 않고 기초과정을 마칠수 있도록 전과정에 걸쳐 단계별로 이론과 실기를 서술한 책이다.

주요 특징을 보면, 초보자라도 이해를 빨리 할 수 있도록 설명을 간단명료하게 했다는 점과 단계적인 예제와 보충설명을 첨가한 점, 복습은 물론, 특히 컴퓨터 경진대회를 준비할 수 있도록 배려한 문제 수록, 문제의 결과를 뒤로 하여 스스로 이해를 한 뒤 예제를 응용할 수 있도록 문제를 배열한 점 등을 들 수 있는데 전반적으로 경험에 의한 결과로서 학습효과를 세심하게 기대했음이 엿보인다.

책의 구성은 6장으로 되어 있고 뒤에는 예약어 해설과 에러메시지, 제어코드가 나온 부록이 있다. PC

새로나온 책

PC 활용을 선도하는 "영진출판사"의 새로운 책들

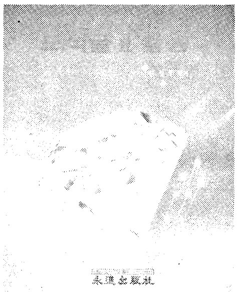


건설 견적업무를 PC로

dBASE III+ 공사내역서 작성 프로그램집

김재율 저 4×6배판 고급양장 값 8,000원

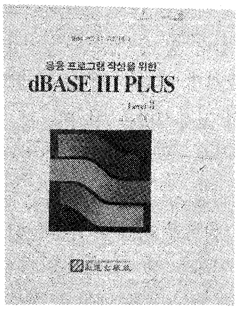
컴퓨터 프로그램 보호법에 의해 과학기술처에 등록된 건설분야 내역서 작성용 프로그램을 소개한 프로그램집.
수시로 단가가 변동하는 자원을 관리하기 쉽고, 내역서가 자원과 일위대가에 별도로 연결되므로 자료의 입력과 수정이 편리하며, 간접비 계산 등 비율 계산도 수시로 변경할 수 있는 장점이 있다. 또한 표준 일위대가표에 새로운 자료를 추가함으로써 기술을 축적시켜 건설 분야의 현대화를 이룰 수 있게 해줄 것이다. 특히 이 프로그램은 PC에서 가장 널리 사용되고 있는 dBASE III PLUS로 작성되어 있고, software도 별도로 판매하고 있어 사용자들에게 더욱 큰 도움이 될 것이다. 이제 견적 업무와의 씨름은 컴퓨터에 맡기고, 현대 기계문명의 위력을 만끽해 보자. Software 1장 ₩10,000



한글 영문 漢文 워드프로세서 보석글 II 활용

편집부 편저 4×6배판 값 4,000원

기능은 다양하지만 배우기가 어려운 보석글 II를 쉽게 설명한 새로운 타입의 보석글 II 활용 도서.
워드프로세서의 기본적인 용어 해설에서부터 깨끗한 출력을 얻기까지, 모든 내용이 화면을 통해 자세하게 설명되어 있다. 보석글 II V2.0의 다양하고 우수한 워드프로세서 기능을 쉽게 이해할 수 있으므로 이 책과 함께라면 당신도 보석글 II를 정복할 수 있을 것이다.



응용 프로그램 작성을 위한 dBASE III PLUS (Level 3)

김경태 저 4×6배판 값 6,000원

dBASE의 기본적인 이해와 프로그래밍의 필수 요소를 다룬 Level 1, 2를 기초로 하여 Level 3에서는 dSALES와 dMAIL 2가지 dBASE application을 다루고 있다. dSALES는 영업사원을 고용하여 대리점 관리와 이에 수반되는 판매 활동을 dBASE 프로그램으로 처리하는 응용 프로그램이고, dMAIL은 dBASE를 사용하여 주소록의 관리와 상용서신의 작성을 처리하는 데 도움이 될 것이다. 부록으로 source code를 첨부하였고, 별도로 program diskette도 판매하고 있다. Software 1장 ₩10,000원



Synthesizer와 컴퓨터 Music

(MIDI 입문과 활용)

이인성 편저 4×6배판 고급양장 값 8,000원

이제 드디어 컴퓨터를 이용하여 자신이 직접 곡을 작곡, 편곡, 연주할 수 있는 컴퓨터 음악 시대가 왔다. 이 책은 악기의 접속에서부터 악보입력, 연주방법까지 사진을 곁들여 알기 쉽게 설명한, 컴퓨터 뮤직에 대한 가장 완벽한 해설서이다. 부록으로 실제 컴퓨터 뮤직을 연주하는 분들을 소개하였고, 카세트 테이프를 제공하여 컴퓨터 음악을 직접 접해볼 수 있게 하였다. 이 책과 함께 컴퓨터 뮤직에 입문하여 청중이 아닌 연주자의 입장에서 컴퓨터의 무궁한 기능을 한껏 즐겨보기 바란다.

초보자를 위한 MS-DOS 入門

김우용 저 4×6배판 값 5,000원

초보자들에게 있어 가장 쉽고 가장 훌륭한 지도교수가 될 DOS 입문서. 본문에서는 DOS의 기초와 활용, 화일 관리, BATCH 화일의 작성과 이용, 하드디스크 사용법 등의 내용을 그림과 화면을 통해 쉽게 설명하였고, 부록에는 DOS 명령어를 수록하여 보다 자연스럽게 MS-DOS에 접근할 수 있도록 하였다. 기초지식에서부터 MS-DOS 활용법에 이르기까지 초보자들이 단계적으로 학습하여 활용할 수 있도록 쉽게 설명하여 독학생에게는 물론 컴퓨터 메이커의 매뉴얼, 학원·학교·회사 등의 교재로도 적합하다.



PC통신 시리즈 II LAN 入門

묵현상·정영일 편저 4×6배판 값 7,500원

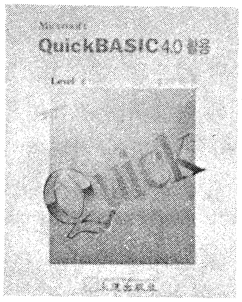
"해설 PC 통신"에 이은 PC 통신 시리즈 후편으로, 이 책에서는 전편의 PC 통신 전반에 관한 이론에 기초하여 LAN 및 관계 소프트웨어, internetworking, 애플리케이션 등 PC LAN에 대해 중점적으로 해설하고 있다. 이 외에도 네트워크 오퍼레이터나 관리자에 대한 가이드라인, 부록으로 PC 데이타 통신의 프로그래밍 기법도 상세히 설명하였다. PC 통신 시리즈인 이들 상하권을 읽고, 당사 발행의 "Crosstalk 매뉴얼"을 참고한다면 데이타 통신과 LAN을 완벽하게 마스터할 수 있을 것이다.



Microsoft 1 QuickBASIC 4.0 활용

장석원 저 4×6배판 값 5,000원

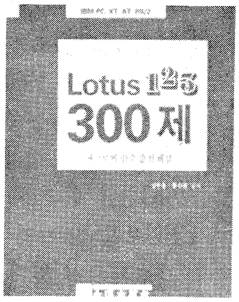
퀵베이직 4.0은 강력한 명령어, 구조적인 체제, 편리한 디버깅 시스템, 사용자 라이브러리 구축 등의 놀라운 기능을 가진 강력한 언어 소프트웨어이다. 이 책은 초보자가 퀵베이직 4.0을 가장 쉽게 이해할 수 있도록 쓰여졌으며, 퀵베이직 통합환경의 완벽한 사용방법, 구조화 프로그래밍 기법, 화일 I/O 기능, 한글 처리의 문제점과 해결방법, 퀵베이직용 ISAM 라이브러리 등을 다루고 있다. 이후에 출간될 실무 중심의 Level II와 함께 여러분이 능숙한 퀵베이직 유저가 될 수 있도록 도와줄 것이다.



단기완성을 위한 Lotus 1-2-3 300제

김우용·홍수현 공저 4×6배판 값 6,000원

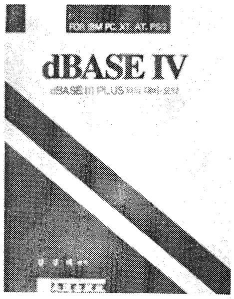
컴퓨터를 통한 사무 자동화를 꿈꾸고 계신 분들께 주저하지 않고 이 책을 권한다. 초보자를 위한 단기 완성편과, 예제를 통한 단계적 학습코스를 거쳐 짧은 시간에 Lotus의 사용법을 스스로 터득할 수 있게 해주고, Lotus 1-2-3의 꽃이라 할 수 있는 매크로를 이용해 기성 프로그램에 못지 않은 막강한 기능을 발휘할 수 있게 해줄 것이다. Lotus 1-2-3 매크로와 함수에의 도전은 여러분에게 업무의 전산화를 통한 새로운 세계를 열어줄 것이다.



dBASE IV (dBASE III PLUS와의 대비 · 요약)

김경태 편저 4×6배판 값 4,000원

이 책은 dBASE III PLUS에 익숙해져 있는 user들에게 dBASE IV의 향상된 기능들을 소개하고 구성을 설명함으로써 이 새로운 버젼에 대한 이해를 돕고자 하였다. dBASE의 기본 지식을 토대삼아 Control center에 의한 일괄처리 menu driven 방식, Development edition package의 지원으로 응용 프로그램 개발의 완벽한 tool을 제공, window · 내장 compile 기능, Graphic software와의 접속 등 dBASE IV의 놀라운 기능들을 쉽게 익힐 수 있을 것이다.

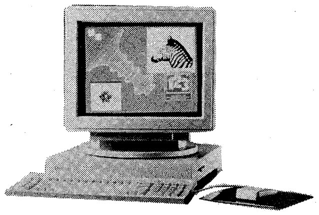


최첨단 도서 전문 출판
永進出版社

서울특별시 용산구 원효로 1가 28-3
창도빌딩 2층
TEL:702-1119(대표) FAX:719-6550

신제품

소형 데스크탑 워크스테이션
SPARC station1



본 제품은 12.5MIPS / 1.4MFLOPS를 실현한 워크스테이션으로 IBM PC 크기의 절반에 지나지 않는 모양을 가진 데스크탑형 PC이다.

SPARC station 1은 UNIX 환경하에 여러가지 작업을 동시에 운영할 수 있도록 성능을 통합하였고 RISC칩을 채택하여 기존 워크스테이션보다 2배 이상의 빠른 처리속도를 가진다.

또 500종 이상의 SPARC웨어 응용프로그램 사용이 가능하고 3개의 확장슬롯이 마더보드에 설계되어 있다. 그리고 ASIC기술에 의해 소형화를 실현하여 CPU보드 크기가 편지지 한장 크기로 간소화되어 있다.

전력 역시 일반 PC에 비해 1/2정도로 절전형이다.

주요 규격은 다음과 같다.

- 프로세서 : SPARC
- 메인메모리 : 16MB
- 인터페이스 : SCSI-2
- I/O포트 : RS-423 2개
- 시스템버스 : S버스, 어드레스버스-32비트, 데이터버스-32비트, 슬롯수-3개
- 키보드 : 94키
- 마우스 : 3버튼
- FDD : 1.44M(3.5")
- HDD : 104MB(2개)

문의처 : 버슨 마스텔러 한국지사
전 화 : (02)782-7151 7

교육용 16비트 XT
SERVER MICRO XT-Ⅱ



본 제품은 10MHz의 i8088-1 고속처리 마이크로프로세서를 장착한 16비트 XT로서, IDP(Intelligent Data Protector) 가 부착되어 있어 패스워드(Password)를 이용하는 2중 안전방식을 갖추고 있다.

또한 기억용량은 640KB를 탑재하고 있고 FDD는 2개의 헤드를 갖춘 더블사이드로 인치당 48트랙을 사용한다. 그리고 각종 정보를 고속으로 처리할 수 있는 0.5MB의 대용량이다.

뿐만 아니라 키보드의 경우는 스트림(Stream)형으로 되어 있고, 입력시에는 매커니컬형식으로 이루어지고 입력을 확인할 수 있는 클릭(Click)기능이 있다.

한글처리 또한 조합형 한글, 완성형 한글, 그리고 KS 2바이트 행정전산망용 한글/한자를 동시에 지원한다.

주요 규격은 다음과 같다.

- 그래픽지원 : 모노그래픽어댑터 EGA, VGA, CGA(옵션), KSC 5601 2바이트(옵션)
- FDD : 5.25"(360KB) 2개
- HDD : 20/30MB(옵션)
- I/O포트 : RS-232C 2개, 1/2 패러럴프린터포트

문의처 : 랩시스템(주)
전 화 : (02)567-7291

랩탑형 PS/2
P70-386



본 P70-386은 IBM PS/2계열로서 8M바이트의 주기억용량을 가지는 랩탑형 컴퓨터이다.

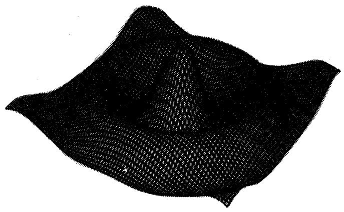
그리고 MCA(마이크로채널 아키텍처)를 채택한 포터블 랩탑컴퓨터로 20MHz 의 i80386 마이크로프로세서를 탑재하였고, 16비트 및 32비트 확장슬롯을 각각 한개씩 제공하고 있으며, 외부기억장치의 케이블 선택기능도 갖고 있다.

특히 16단계의 휘도조절 기능을 갖고 있어 가스 플라즈마 디스플레이가 가능한 것이 특징이다.

주요 규격은 다음과 같다.
- CPU : i80386(20MHz)
- Co-Processor : i80387(20MHz) (옵션)
- 메모리(RAM) : 4~8MB(85ns) (최대 16MB)
- FDD : 1.44MB(3.5") 1개
- HDD : 60MB
- 디스플레이 모드 : VGA, CGA, EGA, MCGA 적합, 640×480픽셀×16 그레이레벨
- 키보드 : 101키
- 공급전원 : 85W
- 크기 및 무게 : 465×305×126㎜ (WHD), 9㎏

문의처 : 한글 아이·비·엠(주)
전 화 : (02)781-6943

수학계산전용 소프트웨어
매시매티카



본 매시매티카는 수치계산 및 기호, 그래픽계산을 완전 자동처리해 주는 수학계산 전용소프트웨어이다.

700개 이상의 함수를 내장한 대화형 소프트웨어로 지수함수와 삼각함수, 행렬처리, 적분은 물론 베셀(Bessel) 함수나 초기화(Hypergeometic)함수, 타원적분(Eliptic Integral) 등의 계산이 가능하고 기호적인 미적분 처리까지 가능하다.

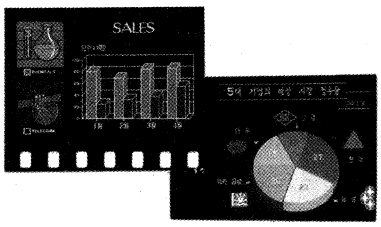
그래픽계산 결과는 2차원이든 3차원이든, 흑백이든 컬러이든 관계없이 함수와 데이터를 플로팅할 수 있고, 출력에 포스트스크립트(PostScript) 표준을 채택하고 있어 해상도에 구애받지 않는 그래픽스를 컴퓨터화면(VDT)이나 레이저 프린터 또는 대형 포스터에 출력시킬 수 있으며 3차원의 색체도면도 가능하다.

특히 이식성이 뛰어나 대형컴퓨터에서 수행하던 복잡한 수학계산을 퍼스널 컴퓨터나 테크니컬 워크스테이션, 퍼스널 수퍼컴퓨터 등에서도 처리가 가능하도록 한 것이 특징이다. 그러나 매킨토시의 경우는 최소한 2MB 이상의 메모리가 필요하다.

그외 기능은 다음과 같다.
- 출력형태 : 소스화일과 결합이 적합한 C와 FOTRAN의 형태, TeX형태 가능
- 구성 : 커널(수학계산 수행)과 프론트엔드(사용자와의 대화)로 구성
- 사용자 스스로 함수 추가 기능

문의처 : 대성산업(주)
전 화 : (02)511-9323

한글컬러그래픽 전문소프트웨어
Graphia



한글컬러그래픽 전문소프트웨어인 Graphia는 한글, 한자, 영문, 그래프 등의 자료를 OHP 필름 또는 35㎜ 슬라이드필름으로 작성하거나 바로 스크린에 출력시켜 볼 수 있어 브리핑도 겸할 수 있다. 그러므로 Graphia는 OHP 필름 또는 35㎜ 슬라이드 필름을 작성하거나 컴퓨터를 이용한 브리핑시스템에 반드시 필요한 한글컬러그래픽 전문소프트웨어이다.

도형작성, 그래프작성, 한글, 한자, 영문 입력이 가능하고 마우스를 이용한 다양한 편집기능-확대, 축소, 스케일, 복사, 입체효과 등-을 가지고 있으며 특히 영상입력장치인 스캐너로 책에 있는 그림을 그대로 옮겨 이용할 수 있어 사용자로 하여금 여러가지 자료를 만들 수 있도록 한다.

Graphia를 운용하기 위해서는 시스템 구성으로 AT 호환기종과 메인메모리 640KB이상, 하드디스크 20MB이상, 플로피디스크 1.2MB이상, EGA 또는 CGA 보드와 모니터, RS-232C 포트 2개 이상, 마우스, 이미지스캐너와 인터페이스 카드 등이 필요하고, OS는 MS-DOS 3.0이상으로 640×350 도트의 화면을 표시한다.

다음은 Graphia의 주요 기능을 나타낸 것이다.
- 글자입력 : 워드프로세서의 기능 수행
  - 영문폰트 : 8×8, 8×16, 16×8, 16×16, 24×24
  - 한글폰트 : 16×16(고딕), 24×24(명조)
- 화면복사 : 다른 그래픽소프트웨어로 작성된 이미지 사용 가능
- 이미지입력 : 이미지스캐너를 사용하여 그림이나 회사 로고 등을 화면에 그대로 표시
- 도형처리 : 점, 직선, 사각형, 원, 다각선, 폐곡선, 무늬사각형, 무늬원 등 처리
- 그림편집 : 크기에 관계없이 확대, 축소, 색칠(1, 2, 3), 회전, 기울림, 입체효과, 이동, 복사, 삭제, 다양한 모양의 붓, 화면 일부의 저장과 읽음 등 처리
- 그림사전의 삽화이용 : 500여 가지의 삽화가 내장되어 있고 이용자가 새로 만들어 등록 가능
- 그래프자동작성 : 데이터의 입력만으로 파이, 막대, 3D막대, 가로막대 등 11가지 형태의 그래프 가능
- SCAN : 라인당 512도트로 6.4㎝의 범위와 200DPI의 분해능, 흑백색과 그레이색표시

문의처 : 픽셀시스템
전 화 : (02)792-4575

●광장●

뭐든지 물어보세요

마우스에서 볼과 센서형의 차이는—

마우스에는 볼과 센서형이 있다고 하는데 실제 사용에서 어떠한 차이가 있는지요? 그리고 버튼도 2개와 3개가 있는 마우스가 있는데 사용상 차이와 문제점은 무엇인가요?

조민행 / 충남 서산시 동문1동 879-9

사용상 차이는 없으나 센서형이 민감하게 작동

마우스에는 메커니컬과 옵티컬, 옵토메커니컬타입이 있는데 매커니컬타입에는 다시 볼과 휠타입이 있습니다.

볼타입은 마우스 바닥에 볼이 돌출되어 있어 볼의 움직임에 따라 두개의 엔코더를 회전시키는데 이를 가로축(X)과 세로축(Y)의 두가지 전기신호로 분해하여 컴퓨터에 전달하는 방식입니다.

휠타입은 두개의 휠이 바닥에 나와 있어 각각의 엔코더를 회전시켜 전달하는 방식이며, 옵티컬타입은 센서형으로 발광다이오드에서 서로 다른 파장의 빛을 발하면 반대편 감광센서가 이를 감지하여 전달하는 방식인데 여기에는 반드시 검정색과 파란색의 격자가 그려진 패드(pad)가 필요합니다.

마지막으로 옵토메커니컬형은 볼타입과 비슷하나 마우스안의 휠에 작은 구멍이 뚫려있어 각각의 양쪽에 발광다이오드와 감광센서가 회전에 따라 빛을 감지하여 전달하는 방식입니다.

이상에서 본 볼과 휠, 그리고 센서마우스는 사용상 차이가 없으나 센서를 이용한 것이 사용상 상당히 민감하게 작동하고 가격면에서도 더 비싸다는 것입니다.

다음은 버튼의 수에 관한 것인데 버튼은 1개에서 3개까지 있습니다. 심지어 16개짜리가 있는데 이것은 디지타이저 등의 특수용도에 사용되는 것입니다.

1개짜리는 매킨토시 컴퓨터에서 사용하는 것으로서 선택과 실행 등의 기능을 하나의 버튼으로 작동시킵니다. 이를 리턴버튼이라고 합니다. 2개짜리는 표준형으로 널리 사용하고 있는 타입으로서 범용캐드에서 많이 이용되고 있습니다. 버튼의 구성은 리턴과 코맨드를 실행하는데 사용됩니다. 3개의 버튼이 있는 마우스는 리턴과 코맨드 기능외에 리드로우(redraw)라는 기능이 추가된 것으로 같은 기능을 복사할 때 사용합니다. 이것은 주로 전문 캐드 등을 이용하고자 할 때 사용됩니다.

이들 3종류는 사용상의 차이는 없지만 3개의 버튼이 있는 마우스의 경우 전문프로그램 등에서 주로 사용됩니다.

마우스의 연결방법에는 9핀, 25핀 코넥터와 PS/2전용, 그리고 카드가 포함된 버스타입이 있어 사용자의 연결 가능 상태에 따라 선택 접속할 수 있습니다. 여기서 9핀은 초기형태로 메인보드에 직접 연결하여 사용하지만 사용중 부주의가 발생할 경우 메인보드에 치명적인 해를 줄 수가 있습니다. 그리고 25핀은 많이 사용하는 시리얼 포트를 이용한 것으로 사용되는 핀은 9핀과 같습니다. 단지 메인보드로 직접 입력되는 것이 아니라 시리얼카드를 통한다는 것입니다.

이말은 위에서 언급한 사용상의 실수로 인한 메인보드상의 치명적인 해가 시리얼카드로 끝날 수 있다는 것입니다. 그러나 요즈음 카드 자체에서 9핀과 25핀을 선택할 수 있는 타입이 나오고 있고, 플로터나 통신카드 등을 이용할 경우 2-시리얼카드나 4-시리얼카드를 사용할 수도 있습니다. 구입이나 사용상의 좀더 자세한 내용은 본지 구입가이드란을 참조하시기 바랍니다.

EXE화일과 COM화일의 차이는

실행화일을 보면 확장명이 EXE인 것과 COM인 것이 있습니다. 모두 실행이 되고 기능도 비슷한데 왜 이렇게 확장자를 2종류로 하고 있는지 모르겠습니다. 무슨 특별한 이유라도 있는지요

이덕준 / 울산시 중구 학산동 13-16

메모리에 배치되는 형태가 다릅니다.

OS(여기서는 MS-DOS)가 실행화일을 주기억 메모리로 로드할 때는 여러가지 관련 정보도 참조하면서 메모리 배치를 하고 있습니다. 일반적으로

메모리는 64KB의 크기를 갖는 세그먼트라는 단위로 구분하여 사용하고 있으므로 임의의 실행화일을 실행시키면, OS는 현재 이용이 가능한 세그먼트를 찾아 프로그램 코드를 로드시킵니다.

이때 모든 세그먼트의 시작부분부터 오프셋번지가 주어지는데 오프셋 0~100H 사이에는 PSP라는 영역이 할당되고 100H 이후부터 실제 프로그램 코드가 들어서게 됩니다.(한예로 어셈블리 프로그램에서 시작부분에 'ORG 100H'라는 명령을 지정하는 이유도 여기에 있습니다.)

이 PSP(Program Segment Prefix)라는 영역에는 명령라인으로 주어지는 각종 파라미터나 엔트리포인트 및 FCB, DTA 등의 정보가 들어가게 되며 대부분 OS에 의해 참조됩니다.

COM화일은 단일 세그먼트를 이용하는 경우로서 PSP영역을 포함하지 않은 채로 저장되었다가 로드될 때는 OS가 자동적으로 100H바이트에 해당하는 PSP영역을 확보해 주므로 코드용량이 적게 되고 모든 세그먼트 레지스터도 오프셋 0로 통일되어 있습니다.

이에 비하여 여러 세그먼트를 참조해야 할 EXE 화일인 경우는 PSP정보 및 자신이 메모리에 배치될 릴로케이션 정보들이 모두 포함된 채로 저장되기 때문에 코드용량이 커지게 되고 로드시에는 DS, ES 레지스터는 오프셋 0로 지정되며 CS, SS는 프로그램 코드가 시작되는 곳에 지정되게 됩니다.

참고로 DOS 유틸리티중 EXE2BIN이라는 소프트웨어가 EXE 형태로 되어 있는 화일을 COM형태로 바꾸어 주고 있으며 데이터 세그먼트나 스택 세그먼트를 별도로 지정하고 있는 경우는 COM화일로의 변환이 불가능하게 됩니다. EXE형태로 만든 화일이 여러 세그먼트를 참조할 수 있다는 것은 그만큼 다양한 기능을 갖는 대형프로그램을 만들 수가 있다는 것을 의미합니다. 반대의 경우로서, COM화일로 변환할 수 있는 작은 화일을 구태여 불필요한 정보까지 포함시킨 EXE화일 형태로 보관한다는 것은 그만큼 기억용량만 낭비할 뿐입니다.

SIMCGA2를 실행시켰을 때 메시지가 화면에 나타나지 않는 이유는

SIMCGA2를 실행시켜 게임을 하려고 배치화일을 만들어 수행시키면 메시지가 화면에 나타나지 않습니다. 그러나 메시지가 나타나는 것으로 여기고 키를 입력하면 실행하는데는 이상이 없습니다. 그 이유가 무엇입니까?

채장국 / 경남 산천군 당성면 관정리 260

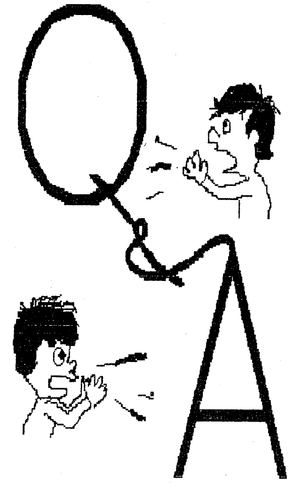
변환방법에서 약간의 차이가 있어 그러합니다.

우선 게임이 어떤 것이었는지 궁금합니다. 허큘리스 보드에서 그러한 현상이 일어나는 경우가 그리 많지 않기 때문입니다. SIMCGA1, SIMCGA2, MAGIC KEY, SUBMAGIC, COLOR 등의 프로그램은 MDA(Mono Display Adapter) 혹은 허큘리스 그래픽보드 등에서 CGA(Color Graphic Adapter) 프로그램을 실행시켜 주는 프로그램입니다. 그런데 MDA와 허큘리스보드는 둘다 그래픽번지가 &B800 : 0000이지만 텍스트페이지의 변환방법 등에서 약간의 차이가 있어 완전한 호환성을 가지지 못합니다.

따라서 MDA를 위한 CGA 에뮬레이션 프로그램을 허큘리스에서 실행시키면 텍스트가 안보이는 등의 증상으로 나타날 수도 있습니다. 그리고 허큘리스를 위한 CGA 에뮬레이션 프로그램에서는 이러한 현상이 더 자주 발생합니다. 만약 가지고 계신 그래픽 보드가 허큘리스 보드라면 SUBMAGIC이나 MAGIC KEY를 추천합니다. (컴퓨터를 취급하는 상인들 마저도 MDA와 허큘리스 보드를 정확하게 구별하지 못하므로 컴퓨터 매뉴얼을 참조하시는 것이 더 정확할 것입니다.)

특히 SUBMAGIC은 ALT-0 키에 의해 그래픽 화면과 텍스트 화면을 토글하므로 텍스트 페이지와 그래픽 페이지를 자동으로 변환시켜 주지 못하는 최악의 경우에도 사용할 수 있기 때문입니다. 그러나SIMCGA나 SUBMAGIC 등을 사용하면 게임을 할 경우에 잔상이 남게 되는데 이러한 경우에는 MAGIC KEY를 사용하십시오. 화면이 작아지기는 해도 훨씬 깨끗한 그래픽 화면을 얻을 수 있을 것입니다.

간단한 아케이드 게임은 텍스트가 안보여도 별문제가 없겠지만 '제트'(F-16, F-18 SIMULATION) 같은 게임은 텍스트상태에서 세팅해 주는 것이 많기 때문에 텍스트가 안보이면 심한 경우 게임을 못할 수도 있습니다.



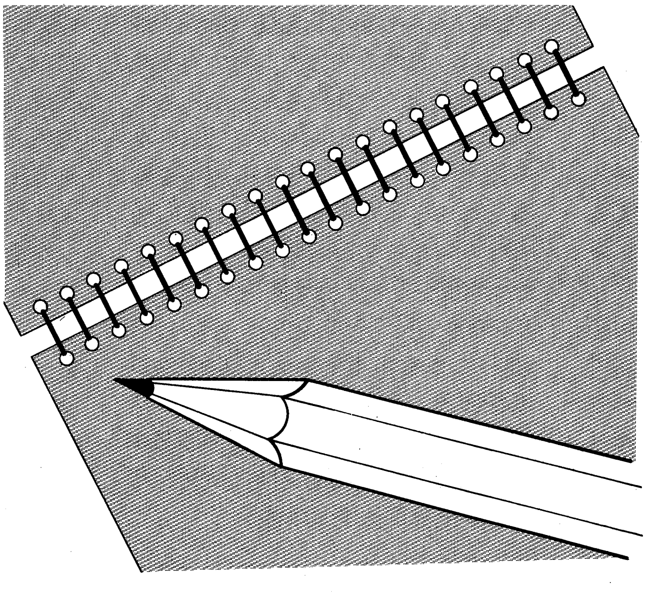
마우스·조이스틱

마우스

개인용 컴퓨터의 사용이 대중화 되기 시작하면서 컴퓨터를 단지 수치처리에 이용하는 것 이외에 그래픽을 처리하는 것과 같은 여러가지 기능들이 요구되었다. 이렇게 복잡하고 다양한 기능들이 요구되면서 컴퓨터의 입력장치인 키보드로부터의 입력작업보다 더욱 편리한 입력장치의 필요성이 대두되었고 이 요구에 부응하여 마우스가 등장하게 되었다.

마우스는 복잡한 용어나 명령을 사용하지 않고 단순히 마우스의 버튼을 누르고 떼는 동작으로 위치를 지정하는 등의 원하는 기능을 수행할 수 있다. 이런 연유로 마우스를 구입하려는 사람들이 많지만 마우스를 어떤 기준에 의해 선택해야 하는지는 잘 알려지지 않고 있다.

이에 본지에서는 사용자의 구입에 도움이 되고자 마우스에 관하여 정리하여 보았다.



(1) 마우스의 종류

마우스는 신호를 전송하는 방법에 따라 다음과 같이 4가지로 분류된다.

① 메카니컬 마우스(Mechanical mouse)

보통 볼 마우스라 불리우며 마우스가 회전하는 방향에 따라 바닥의 볼이 움직여 볼과 연결된 2개의 엔코더를 회전시키고, 엔코더는 볼의 회전을 X축과 Y축의 전기신호로 바꾸어 움직인 만큼의 위치를 파악하여 컴퓨터에 데이터를 전달한다.

② 옵토메카니컬 마우스(Opto-mechanical mouse)

메카니컬 마우스와 기본적인 동작원리는 같으나 기계적 접촉대신 발광 다이오드가 발사한 빛을 감광센서가 읽어 들여 데이터를 전달한다.

③ 옵티컬 마우스(Optical mouse)

pad라는 금속판에 서로 다른 파장의 빛을 발광 다이오드로 쏘아 감광센서가 이것을 감지하여 컴퓨터에 데이터를 전달한다.

④ 휠 마우스(Wheel mouse)

볼의 회전이나 빛을 사용하는 대신에 플라스틱 휠을 사용하여 휠이 마우스의 움직임을 감지하여 직접 엔코더를 움직인다.

또한 마우스는 가지고 있는 버튼 수에 따라 1버튼, 2버튼, 3버튼으로 나누고 있는데 1버튼은 코맨드 버튼, 2버튼은 리턴과 코맨드 버튼, 3버튼은 리턴과 코맨드 버튼, 리드로우(Redraw)의 기능을 지원하고 있다.

(2) 마우스 구입 및 취급시 주의사항

마우스는 닥터할로우, 페인트브러쉬와 같은 그래픽 뿐만 아니라 디베이스, 로터스 1-2-3, 워드스타, 오토캐드 등 사용분야가 광범위하다. 그러나 모든 응용 프로그램에서 마우스를 지원하는 것이 아니므로 마우스가 지원되는 소프트웨어인지 살펴보기 바란다.

마우스를 선택하려면 우선 앞에서 설명한 것처럼 소프트웨어에 관한 사항을 점검해야 하는데 우선 자신의 시스템과 연결이 가능한지 조사한다. 그리고 어느 정도의 소프트웨어를 지원하며 호환성은 어떻게 이루어지는지도 알아보아야 한다.

또한 어떤 방식으로 신호를 전달하며 인터페이스는 어떻게 하는지, 몇개의 버튼을 사용해야 하며 케이블의 길이는 어느 정도인지, 충격에 견디는 내구력은 어느 정도인지(메카니컬 마우스를 구입할 때는 볼의 크기가 크고 무게가 무거운 것을 고르는 것이 좋다), 지원하는 해상도는 얼마인지(보통 200DPI 정도가 가장 많이 사용된다), 사후 처리는 어디까지 가능하며 그 기간은 얼마인지 체크하기 바란다. 무엇보다 가격대 성능비를 조사하여야 할 것이다.

마우스를 구입하는 것도 중요하지만 그 마우스의 가치를 높이는 것은 얼마만큼 잘 활용하며 어떻게 다루느냐에 달려 있

는 것이다. 마우스를 구입하고 나서 사용할 때는 어떤 마우스이건 갑작스런 충격을 가해서는 않된다. 특히 옵티컬 마우스는 패드가 섬세한 금속판으로 구성되어 있는데 이 패드가 약간의 손상이라도 입게 되면 마우스가 제대로 동작하지도 않을 뿐더러 복구할 방법이 없고 약간의 충격에도 센서에 이상이 있을 수 있으므로 특히 주의해야 한다.

볼타입의 메카니컬 마우스는 보통전용의 패드가 없이도 사용할 수 있어 기계적인 마모와 볼에 먼지나 이물질이 낄 수 있으므로 주기적으로 볼을 청소하여 청결을 유지하기 바란다.

(3) 인터페이스 방법

마우스를 컴퓨터에 접속시키는 방법은 다음과 같이 4가지가 있다.

① 버스 인터페이스

사용하는 컴퓨터의 본체의 확장슬롯에 RS-232C를 설치하여 연결하는 방법으로 본체에 있는 시리얼 포트에 이미 다른 주변기기가 연결되어 있을 경우에 사용하는 방법이다.

② 시리얼 인터페이스

RS-232C 포트 중의 COM1이나 COM2 중의 하나에 연결시키는 방법으로 가장 일반적인 인터페이스 방법이다.

③ 인포트 인터페이스

컴퓨터의 메인보드에 코넥터를 내장하여 직접 접속하는 방법이다. 컴퓨터에 효과적인 그래픽 입력장치를 지원하기 위한 인터페이스 방법이다.

현재는 인포트 인터페이스 형태로 변화하고 있는데 이는 컴퓨터에서 그래픽 기능을 요구하는 영역이 확대되고, 소프트웨어의 호환성을 높이기 위함이다. 이것은 마우스가 선택적인 것에서 기본적으로 있어야 하는 주변장치로 변환되고 있음을 보여준다.

(4) 마우스의 기동

보통 XT에서는 25핀 코넥터, AT에서는 9핀 코넥터, PS/2에서는 6핀 코넥터를 사용한다. 따라서 구입하고자 하는 마우스가 자신의 시스템과 연결이 가능한지를 확인하여야 한다.

그 다음 플로피 디스크 드라이브나 하드디스크에 마우스를 지원하는 소프트웨어인 마우스 드라이버를 설치한다. 프로그램을 SET UP시킬 때 가끔 마우스의 선택항목에 자신이 가지고 있는 마우스가 선택메뉴에 없어 당황하게 되는데 이럴 경우 거의 모든 마우스가 호환성을 이루고 있으므로 프로그램이 지원하는 마우스를 선택하면 된다.

마우스를 사용하기 위해서는 다음과 같은 기초동작에 익숙해야 할 것이다.

① 위치지정(Pointing)

마우스를 이용하여 화면상에 커서의 위치를 지정한다.

② 버튼누름(Clicking)

마우스의 버튼을 순간적으로 눌렀다가 놓는 것을 말한다. 이때 화면의 커서 위치에 관계되는 명령을 주도록 한다.

③ 누른 채로 이동(Dragging)

마우스 버튼을 누른 상태에서 마우스를 이동하는 것을 말한다.

조이스틱

컴퓨터 붐이 맨처음 일기 시작했을 때 컴퓨터 주변기기 중 조이스틱의 인기는 매우 높았다. 마치 게임기의 일종같이 인식되어 많은 사람(특히 애플을 다루었던)에게 선호되었다. 대부분 외국의 제품을 수입하여 판매하고 있는데 그치므로 조이스틱은 그 이름이 알려져 있는 만큼 속사정(?)은 알려지지 않고 있다. 이런 의미에서 본지에서는 조이스틱의 원리와 인터페이스, 사용법, 간단한 테스트 방법, 구입시 체크사항에 관하여 조사하였다.

(1) 조이스틱의 원리

조이스틱은 그래픽 프로그램이나 게임을 실행시킬 때 커서의 역할을 담당하고 있다. 조이스틱은 각사마다 약간씩 다르지만 대체로 IBM과 애플의 선택 스위치, 자동발사 스위치, X, Y축 조절과 발사버튼 등으로 구성되어 있다.

조이스틱의 원리는 0-100KΩ의 볼륨 2개를 조합하여 각기 상하 좌우로 나누어 그 저항값으로 위치를 계산한다. 즉, 좌우의 경우 50K를 중심으로 하여 0KΩ의 저항값에서는 맨우측, 100KΩ의 저항값에서는 맨좌측으로 움직인다. 최근에는 저항의 정밀도가 높아져 조이스틱이 떨리는 현상은 많이 감소되었다.

(2) 인터페이스

조이스틱을 본체와 연결하려면 별도의 어댑터가 필요한데 이 어댑터는 포트수에 따라 다르게 구분되며 보통 1~2개로 구분하고 있다. 만약 시스템에 G7이나 멀티 I/O카드가 장착되어 있다면 조이스틱 어댑터를 따로 구입할 필요가 없다.

현재 시판되고 있는 조이스틱은 유원상사의 KC3, 언더우드전자의 MASTER, 대만제 수입품인 QUICK SHOT 등이 주류를 이루고 있다. 각사의 제품의 성능이나 기능은 거의 유사하지만 연속발사 기능을 추가하거나 애플과 IBM PC 호환기종과 호환성이 있는 등 회사마다 약간의 차이가 있다.

(3) 조이스틱 사용법

조이스틱을 사용하려면 우선 자신의 기종(애플인지 IBM PC인지)에 맞추어 선택스위치를 조절한다. 보통 KC3와 MASTER는 하단에 QUICK SHOT는 상단에 선택스위치가 있다.

그 다음은 프로그램에 따라 중심점의 이동을 조절하는 X, Y축을 조절해야 한다. 이 X, Y축의 조절은 스틱의 상단에 장치되어 있으며 X축은 좌우로 커서를 움직여 원하는 위치에 맞추고, Y축은 상하로 커서를 움직여 조절한다.

조이스틱에서는 게임을 제외한 그래픽 팩키지에서 조이스틱의 떨림을 최대한으로 줄이는 역할을 담당하는 패들이 장착되어 있는데 이 패들을 사용하게 되면 스틱을 자동적으로 중앙의 원위치에 가게하는 Auto Center 기능이 제거된다. 패들에서 X, Y축의 조절은 스틱의 밑면에 장치되어 있는데 손잡이를 그림 1의 a와 같이 놓고 X2, Y2 스위치를 오프상태로 하고, 다시 손잡이를 그림 1의 b와 같이 X1, Y1 스위치를 오프상태로 하면 된다. 여기서 주의할 것은 손잡이를 대각선 방향으로 한 후 스위치를 조절해야 하며 X, Y 스위치를 그냥 조절하면 손잡이의 스틱이 손상을 입게 된다.

(4) 조이스틱의 테스트

조이스틱을 구매하고서는 어떠한 사항을 체크해야 할지 궁금할 것이다. 이럴 때는 다음과

〔그림 1〕



(a)　　(b)

같이 간단하게 베이직에서 다음과 같이 테스트해 보라.

```
10 PRINT STICK(0) ; STICK(1)
   ; STRIG(1) ; STRIG(4)
20 GOTO 10
```

여기서 STICK(0)는 조이스틱의 좌우값으로 0에서 255까지 변하는데 평균값은 127이나 128이다. STRIG(1)과 STRIG(4)는 조이스틱의 0번과 1번 버튼을 눌렀을 경우 −1 값을 가진다.

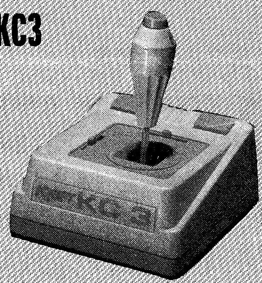
(5) 구입시 체크사항

조이스틱은 마우스의 대용으로도 사용할 수 있는 구매력있는 주변기기로 구입할 때는 우선 저항값의 유동성이 적고, A/S가 어디까지 가능한가 체크해야 하며 어댑터와 조이스틱의 연결(타사의 제품을 사용하게 되면 스틱이 잘 움직이지 않을 경우도 있다)이 잘되는지 점검하고 버튼의 위치가 조절하기 편안한 위치에 있는지도 살펴야 한다.

그리고 조이스틱을 구입하고 나서 사용할 때는 무리한 힘을 주거나 부주의하게 떨어뜨리지 않도록 한다. 강한 충격은 조이스틱의 조립에 손상을 입힐 수 있다.

조이스틱

KC3



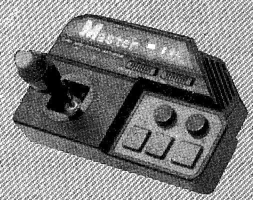
- 공급처 : 유원상사
  (02)273-7729
- 애플 Ⅱgs, Ⅱe, Ⅱc, IBM XT/AT 호환
- Free-floating 모드
- X, Y축 조절가능
- Ayto center 기능 내장
- 발사버튼 : 3개
- 가격 : 1,7000원(VAT 별도)

Quick shot



- 공급처 : 아프로만
  (02)279-8818
- 애플 Ⅱ+, IBM XT/AT 호환
- 자동발사기능
- X, Y축 조절가능
- Auto Center기능
- 발사버튼 : 2개
- 가격 : 1,5000원(VAT 별도)

Master Ⅱ



- 공급처 : 아이콤
  (02)417-2375
- 애플 Ⅱ+, Ⅱe, Ⅱgs, IBM XT/AT 호환
- 자동발사기능
- X, Y축 조절가능
- 발사버튼 : 3개
- Auto Center 기능
- 가격 : 1,8000원(VAT 별도)

마우스

MS 마우스

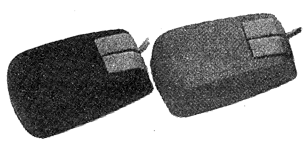


- 공급처 : (주)마이크로소프트
  (02)782-3011
- 메카니컬 마우스
- 2버튼 방식
- 해상도 : 200DPI
- 인터페이스 : 버스인터페이스, 시리얼 인터페이스, 인포트 인터페이스
- 가격 : 12만원(VAT 별도)

PC 마우스 Ⅱ

- 공급처 : 대일전자
  (02)272-5165
- 메카니컬 마우스
- 2버튼 방식
- 인터페이스 : 시리얼인터페이스
- 가격 : 10만원(VAT 별도)

COLOR-MOUSE



- 공급처 : 벤엘전자
  (02)275-2887
- 메카니컬마우스
- 2버튼 방식
- 해상도 : 250DPI
- 인터페이스 : 버스 인터페이스
- 가격 : 4만원(VAT 별도)

SUPER Ⅴ

- 공급처 : 삼정마이컴
  (02)273-8258
- 옵티컬 마우스
- 3버튼 방식
- 해상도 : 250DPI
- 인터페이스 : 시리얼인터페이스
- 가격 : 10만원(VAT 별도)

보내는 사람

□□□-□□□

우편요금
수취인후납부담
발송유효기간
1989. 2. 1～1991. 12. 31
서울 체신청
승인119호

월간 퍼스널컴퓨터 하이테크

서울특별시 용산구 갈월동 17

퍼스널컴퓨터 편집부앞

140-150

애독자페이지

관심있는 분야		게재를 원하는 기사	

□ 컴퓨터에 관한 알고 싶은 질문

□ 본지에 바라는 의견

정기구독신청서

구독신청방법	□ 은행지로 □ 우체국 대체구좌 □ 소액환 □ 기타		
구독기간	년 월호부터 년 월호까지	구독신청일자	

□ 받으실 분의 이름과 주소(우편번호)

☎ (—)

※ 이곳에 풀칠을 하여 뒷면 수신란이 나오게 접어서 우체통에 넣어 주세요(우표부착 불필요).

애독자 설문지

책의 편집에 반영하고자 하오니 성의껏 적어서 보내주시기 바랍니다.

작성자	이름		나이		직장		성별
	주소	☎ (　　–　　)					
사용기종	8비트		16비트		32비트		
	주변기기	□ 모니터(컬러, 단색)　□ 드라이브(HDD, FDD)　□ 프린터(9핀, 24핀, 레이저)　□ 인터페이스카드(　　)					
	사용용도	□ 취미　□ 업무　□ 교육　□ 학습　□ 기타			사용경력		
	사용언어	□ 베이직　□ C　□ 파스칼　□ 기계어　□ 기타					
	사용분야				사용 소프트웨어		
책구입방법	구입방법	□ 정기구독　□ 서점구입(　　서점)　□ 증정　□ 기타					
	구입동기	□ 정기구독　□ 필요한 내용이 있어서　□ 주위에서 권해서　□ 기타					
	본지외 구독잡지	□ 없다　□ 있다(잡지명 :　　)			원하는 발행일자		
	애로사항	□ 가격이 비싸다　□ 종이 질이 나쁘다　□ 인쇄상태가 나쁘다　□ 기타					
	□ 본지 구독을 추천하고 싶은 사람이 있다면(이름과 주소)　☎ (　　–　　)						
이번달에 좋았던 기사(2가지)	제목				제목		
	□ 이유				□ 이유		
이번달에 싫었던 기사(2가지)	제목				제목		
	□ 이유				□ 이유		

자르는 선

월간 퍼스널컴퓨터의 정기구독자에게는 아래와 같은 특전이 마련되어 있습니다.

—정기구독자는 본사가 보유한 각종 정보 및 자료를 무료로 배부받으실 수 있읍니다.
—본지에 게재되는 각종 프로그램을 디스켓에 담아 무료로 보내드립니다.
—본지 및 본지의 자매지(월간 전자과학) 주최로 개최되는 세미나 및 각종 행사에 무료로 참석하실 수 있읍니다.
—본지에서 행하는 각종 부대사업에 할인혜택이 주어집니다.
—구독기간중 책 가격이 인상되더라도 종전가격으로 구독이 가능합니다.

● 구독신청은

—시중은행에서 지로용지를 이용하시거나 우체국에서 대체구좌 또는 소액환으로 송금하시면 됩니다.
은행 지로번호 : 3000667
우체국 대체구좌번호 : 010256—31—0526772

● 정기구독료는

— 1 년 구독시 30,000원
— 6개월 구독시 15,000원

● 구독신청시 주의사항은

—구독기간과 주소(우편번호), 전화번호와 받으실 분의 성명을 정확히 기입하시기 바랍니다.
—배달사고나 기타 변동사항은 본지 영업부로 즉시 연락하시기 바랍니다. 신고전의 분실사고는 책임지지 않습니다.

● 정기구독 문의처는

—서울시 용산구 갈월동 17 ㊇ 140-150
월간 퍼스널컴퓨터 영업부
전 화 : (02)754—1044(대표)
FAX : 757—1049

▒ 전국총판 ▒

●서울●	종로, 중구, 마포, 용산	명문사	774—6314
	중랑, 성동, 동대문	합동사	242—8772
	노원, 성북, 도봉	진원사	996—2443
	강남, 강동, 송파	광일사	415—7330
	관악, 동작	관악사	889—4504
	서대문, 은평	형제사	304—9462
	영등포, 구로	영도사	613—5381
	강서, 양천	연우사	865—8239
●부산●		종합기술서적	462—2431
		영광도서전시관	805—9500
		북부총판	805—9677
		남부총판	756—2727
●대구●		청운서림	23—7335
		한일서적	23—7101
		학원사	22—0336
●인천●		동양서점	73—3384
●광주●		삼복서점	22—0258
●경기●	부천	대경서점	62—3672
	안양, 안산	사랑글방	57—1916
	수원	동아서점	5—6436
	성남	영광서점	2—6532
●강원●	원주	원주종로서적	42—2943
	삼척	영동사	72—6060
	강릉	삼일사	2—2314
	춘천	대진서점	54—5148
●충북●	충주	문학사	2—2814
	청주	화문당	55—0503
●충남●	천안	동방서적공사	63—3314
	대전	동양서점	22—1538
●경북●	포항	경북서점	2—2717
	구미	중앙서점	52—7915
●경남●	진주	동아서점	42—3768
	진해	학애서점	2—2523
	울산	우정서점	43—6003
	마산	기술서적	56—1417
●전북●	전주	홍지서점	6—5311
	군산	동원서점	42—3776
●전남●	목포	문화당	5—3086
	여수	대양서점	62—2111
●제주●		광양서적	53—5077



□ 프로그램 제공안내 □

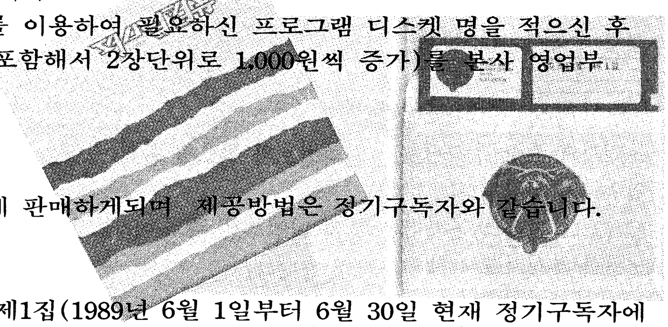
컴퓨터의 보급이 늘어남에 따라 이제는 어디에서나 쉽게 접할 수 있을만큼 대중화되었으며 컴퓨터란 용어자체를 자연스럽게 대화속에 끌어들일 수 있게 되었습니다. 그러나 아직까지는 활용할 수 있는 소프트웨어가 부족하다고 할 수 있습니다. 저희 퍼스널컴퓨터에서는 이러한 분들을 위하여 매달 책에 게재되는 프로그램을 디스켓에 담아 제공하고자 합니다.

□ 프로그램 제공방법

●정기 구독자의 경우

–직접 찾아오시면 본사 영업부 디스켓 담당직원이 정기구독자인지를 확인(주민등록증이나 기타 증명서 필요) 한후 프로그램 디스켓 1장당 1,500원씩에 판매하게 됩니다.

–지방에 계신 분은 본지 뒷면에 있는 은행지로 용지를 이용하여 필요하신 프로그램 디스켓 명을 적으신 후 디스켓 구입비(1장당 1,500원씩 증가)와 송료(자켓 포함해서 2장단위로 1,000원씩 증가)를 본사 영업부 프로그램 디스켓 담당자 앞으로 보내주시면 됩니다.



●일반 독자의 경우

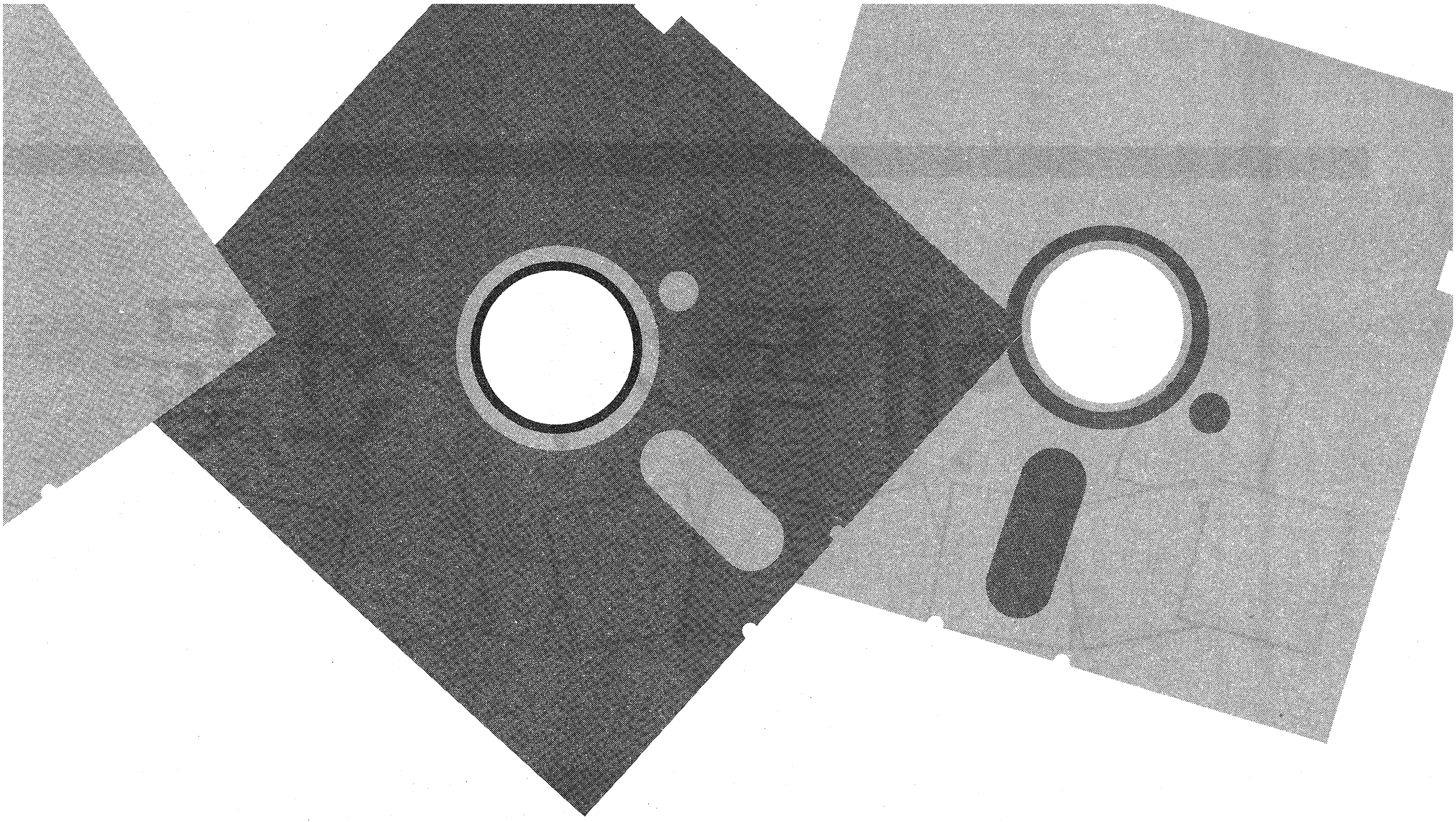
–일반 독자의 경우 프로그램 디스켓 1장당 4,000원씩에 판매하게되며 제공방법은 정기구독자와 같습니다.

□ 알림

정기구독자에게 무료로 보내드리는 프로그램 디스켓 제1집(1989년 6월 1일부터 6월 30일 현재 정기구독자에 한하여 무료로 보내드림)은 당초 6월중으로 전부 보내드릴 예정이었으나 업무량이 폭주해 작업에 차질이 생기는 바 일부 정기구독자의 경우 조금 늦게 도착할 예정입니다. 이점 널리 양해해 주시기 바라며 프로그램 디스켓 제2집은 1집과 마찬가지로 프로그램이 모이는 데로 정기구독자에 한하여 무료로 보내드리겠습니다.

□ 문의처

주소 : 서울시 용산구 갈월동 17 세기빌딩 3층 월간 퍼스널컴퓨터 영업부 디스켓 담당자앞
전화 : (02)754–1044(대표) FAX : (02)757–1049



프로그램 디스켓

▨ 제1집

디스크 논리구조를 이용한 화일복구(88년 12월호 p74)
플로피디스크상에서 실수로 지워진 화일들을 완벽하게 복구시켜 준다.

LX-800의 버그(89년 3월호 p160)
LX-800에서 그래픽을 하드카피할때 생기는 버그를 제거해준다.

스피디 3의 제작(89년 2월호 p166, 3월호 p192)
마이크로 로보트의 소스 프로그램

360KB이상의 데이터를 복사하려면(89년 4월호 p54)
360KB 이상의 데이터를 분리하여 저장할 수 있게 해준다.

레코드판 관리 프로그램(89년 4월호 p94)
200개의 곡명을 수록할 수 있으며 원하는 형태별로 찾아볼 수도 있다.

미분이 가능한 계산기 프로그램(89년 4월호 p110)
공학에서 자주 사용되는 복잡한 계산을 처리할 뿐만 아니라 미분도 가능하다.

그래픽 애니메이터(89년 4월호 p176)
20개의 프레임에 연속적인 이미지를 그린 다음 빠른 속도로 출력시켜 움직임을 표현한다.

QUICKEY.COM(89년 4월호 p200)
램상주 유틸리티로서 키보드의 감도를 빠르게 해준다.

교육용 프로그램(89년 5월호 p76)
물리학습용 프로그램으로 교육용 프로그램의 구성은 어떻게 되어 있나를 알 수 있다.

전화번호부 관리 프로그램(89년 5월호 p88)
디베이스로 간단히 작성한 프로그램으로 전화번호와 주소를 입력하여 검색하거나 우편용 레이블 등을 작성할 수 있다.

□ 주의
각 화일들은 아카이브화 되어 있기 때문에 확장시킬 때 용량이 모자랄 수도 있으므로 이점에 유의하여야 하며 프로그램을 실행시키기 전에 반드시 README 화일을 불러 확인한다.

유틸리티

▨ 마우스 활용 유틸리티

마우스 드라이버 툴키트(89년 5월호 p94)
마우스 드라이버 프로그램으로 각자가 손쉽게 자기용도에 맞게 응용할 수 있는 프로그램 툴이다.

그래픽 에디터 SEM-DRAW(89년 5월호 p112)
여러가지 그래픽에 필요한 루틴들을 첨가하여 마우스를 이용해 손쉽게 그래픽을 처리할 수 있도록 한 에디터이다.

▨ BBUN(89년 6월호 p100)
터보 에디터의 장점을 모두 갖춘 새로운 에디터

▨ 취업정보 안내 프로그램(89년 6월호 p214)
퍼스널 컴퓨터를 이용하여 취업정보안내에 관한 각종 자료들을 데이터베이스화하여 작성한 프로그램

공개 소프트웨어

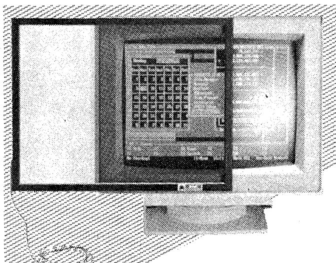
▨ PC매거진(89년 3월호 p200)
KEY-FAKE.COM을 비롯한 41개의 유용한 유틸리티가 담겨 있다.

▨ DX.COM(89년 5월호 p198)
디렉토리 관리 기능을 가진 DX.COM 등 여러가지 다양한 프로그램들이 수록되어 있다.

□ 알림
독자 여러분의 요청에 따라 월별로도 프로그램을 수록하여 제공합니다. 참고로 89년 6월 프로그램 디스켓에는 아래와 같은 내용들이 담겨 있습니다.

TPRINT
NKP에서의 윈도우 구현
램 디스크
음성응답장치
IO.SYS의 분석
명함관리 프로그램

애독자 선물



3명

폴리에스터에 특수 구리도금과
아크릴 수지로 가공 처리된
첨단기술의 V.D.T 증후군
방지용 컴퓨터 보안기이다.

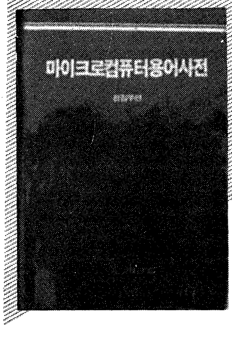
삼종실업 제공
TEL : 784-6931



5명

컴퓨터에 전문지식이 없는 사람도
쉽게 접근할 수 있도록
꾸며진 좋은 지침서이다.

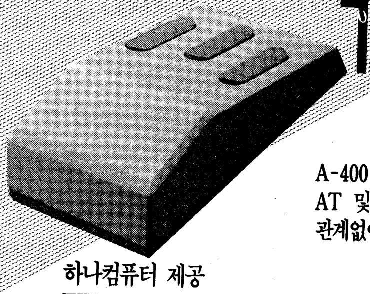
영진출판사 제공
TEL : 719-1119



5명

누구나가 한권씩은 지니고 있어야
하는 컴퓨터 용어사전으로 비교적
자세히 설명되어 있다.

오므사 제공
TEL : 754-1044



1명

A-400 마우스는 IBM PC/XT/
AT 및 호환기종이면 기종에
관계없이 사용이 가능하다.

하나컴퓨터 제공
TEL : 717-4894



3명

특수 산화물 코딩 처리된
에러율 0%의 디스켓이다.

한국쓰리엠 제공
TEL : 785-4567



5명

미국 UL규격을 인정받은 제품으로
아크릴 시트로 특수가공되어
선명한 화면을 유지시켜준다.

화신전자 제공
TEL : 733-3567

대잔치



SKC 제공
TEL : 756 - 5151

균일한 자성체의 배양과 성장으로
도막의 균일도 및 기록신호의
안정도를 극대화하였다.

월간 퍼스널컴퓨터는
독자 여러분의 의견을 소중히 하여
책의 편집에 반영하고자 합니다.
애독자 설문지를 사용하여
질의내용에 성의껏 답하시어
15일까지 보내주시면 27명을 뽑아
선물을 드립니다.

※ 서울 거주자는 7월31일까지 신분증을 지참하시고 직접 본사로 나오시면 해당 상품을 드리며, 지방에 계신 분은 우송해 드리겠습니다.
기간은 7월31일까지이며, 이후는 무효로 처리합니다.

※ 애독자 설문지는 233페이지에 있습니다.

애독자 선물대잔치
6월호 당첨자 명단

● 컴퓨터 보안기

문 웅 서울 강동구 둔천동 30-4 현대 APT 14-708
손순옥 서울 구로구 독산 본동 983-12 7/11
이흥규 충북 청주시 운천동 1635번지 33/2

● MS-DOS 입문

김정욱 부산시 사하구 괴정4동 553-16 10/3
김종호 강원도 삼척시 남양동 22-13 8/3
박종희 서울 동작구 신대방2동 361-26 9/4
임충빈 대전시 서구 갈마동 동산APT 4-202
정하영 인천시 중구 신흥동 3가 7번지 안국APT 2-308

● 디스켓 박스(10장들이)

김정택 경기도 안양시 석수2동 럭키APT 6-1404
윤병목 서울 도봉구 공릉1동 502-9 13/9
임은빈 서울 서대문구 신촌동 연세대학교 전기과 자동화공학연구실
임호찬 경남대학교 전자계산학과 3학년
황문상 부산시 영도구 남항동 2가 197번지 14/5

● 마우스

성은아 경기도 남양주군 진접읍 오남리 진주APT 104-102

● 디스켓 자켓(4 장들이)

신 권 충북 청주시 북문로 2가 92-2
엄외숙 경남 울산시 중구 우정동 278-48 18/1
이창훈 부산시 동래구 낙민동 172 동래한양맨션 2-701

● 컴퓨터 용어사전

김형욱 강원도 춘천시 효자2동 397-16 9/4
박지훈 부산시 북구 덕천1동 779번지 은마APT 7-201
유명배 인천시 남구 숭의1동 342 3/1
정해석 대구시 달서구 성당2동 392-50 29/2
황종율 대구시 서구 비산7동 1282-3 16/5

● 화신컴퓨터 보안기

정재은 서울 서초구 반포1동 715-5
박희섭 서울 은평구 응암1동 103-44
백인성 서울 영등포구 여의도동 하남BD 제일정밀연구소 Banking team
이두원 서울 은평구 응암3동 584-16 7/6
장지영 서울 영등포구 신길동 신길 우성1차APT 107-407

책을 마감하고



● 이것 저것 생각이 많은 달이었습니다. 엎치락 뒤치락대며 아침 저녁으로 줏대없이 바뀌는 생각속에서 우뚝서기 위해 무던히도 애썼던 달이었습니다.

그뿐인가요 불난집에 부채질하듯 무정한 S필자는 말없이 종적을 감추고, 알맞은 필자를 만나기 위해 겪어야 했던 우여곡절이 줄을 타는 광대를 보는 심정이었다면 독자여러분들은 아실랑가요.

그러나 그 와중에서도 성의껏 도와주신 G사의 K님 짧은 시간 많은 분량 감당하시느라 힘드셨습니다. 그리고 D사의 J님께 또한 감사드립니다. 〈김현〉

● 그냥 스쳐지나는 사람들이 저리 많기에
만남이 소중하게 여겨집니다.
헤어짐 뒤의 만남이 그 빈자리를 채워주기에 더 소중한가 봅니다.
이달에 만난이들–
반가워요.! 〈美賢〉

● 교육용 PC 16비트로 선정, 각계의 다채로운 반응, 정보화 시대, 컴퓨터 마인드확산, 퍼스널컴퓨터 7월호, 디베이스강좌 마감. 이렇게 저렇게 89년의 절반을 살라먹고 이제 다시 회생하는 마음으로 남은 절반을 좋은 퍼스널컴퓨터지를 위하여 더욱 좋은 동반자들과 함께 크게도 작게도 바라지 않는 소망으로 보낼 것을 확신합니다. 〈둘리〉

● 여덟!
모처럼
학교에서 열심히 게기고(?) 있을 은아가 생각난다.
후배인 주제에 되려 내게 격려를 해주고 용기를 주던, 치마 안입기로 유명한 녀석이다.
목석같은 이…
돌같이 무정하더니 요번 웬 바람이 불어 전화를 했지?
몸건강 하냐구?
밥은 잘 먹느냐구?
그래 난 잘 있다! 그렇지만 내일까지 원고 못 끝내면 한강물에 들어가야 할 판이라구!
한강 물귀신이 밥 잘먹는거 하구 무슨 상관이 있니? 〈榮〉

● 너무 부족한 상태에서 한달을 달려왔습니다. 배울 것도 너무 많고, 할 일도 너무 많습니다. 하나를 끝내면서, 더큰 무게로 다가오는 책임감이 어깨를 누릅니다. 그저 열심히 해야 할 밖에는… 〈규화〉

● 벌써 반년이 지났다, 아니 89년이.
이제 입사한지 겨우 두달(횟수로), 정확히 한달+? 일.

장미의 계절도 멀리하고 이제 장마와 푹푹찌는 삼복더위가 기다리고 있다. 빨빨거리고 돌아 다닐 생각하니 벌써 아찔하기만 하다.

아니지 먼저 이번호가 잘 나왔는지 봐야겠다. 교정을 본다고 보았는데 처음이라 그런지 아주 굉장히 서툴다. 다음호는 아주 쬐끔만 실수하도록 진짜 빨빨거리고 돌아다녀야겠다. 잘해 보자고(독백). 〈榮珠〉

● 드디어 책이 나오게 됐어요.
이제부터 다음 책을 위해 또 뛰어야 겠어요. 바쁘게 살아갈 때가 가장 좋은 시절이라지요. 아마 〈恩英〉

● 시작–,
에서 벗어나
조금씩 윤곽을 잡아가고 있는(회사업무 및 기타) 요즘,
그렇기에 더욱 기쁨이 큰 나날들입니다.
자신을 가꾸며, 기쁨을 가지고 생활할 때 밝음의 빛은 자신의 곁에 항상 있으리라 믿습니다. 더불어 그 기쁨이 다른 사람들에게 영향을 주는 것이라면 더욱 빛나겠지요. 〈경〉

● 흔히 '난 너무 늦어서……', '난 너무 나이가 많아서……' 라는 말에 얽메여 꿈과 희망을 좌절시킵니다.
허나 중요한 것은
'언제'가 아니고 '무엇'임을 알아야 하리라 봅니다.
지금도 늦지 않았습니다. 시작해 보세요.
지금 당장,
그게 어렵다면, 연필이라도 들어 긁적임부터 시작해 보시지요.
가슴이 벅차오르지 않습니까.!
"중요한 것은
'언제'가 아니고
'무엇'임을 잊지 마세요." 〈花〉